

**Leksell Gamma Knife Treatment Planning via
Kernel Regression Data Mining Initialization and
Genetic Algorithm Optimization**

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Joseph Michael Bowling

May 2012

Copyright © 2012 by Joseph Bowling

All Rights Reserved

Dedication

This dissertation is dedicated to my family, who truly are a gift from God. My loving wife and best friend Sara whose companionship has made the completion of this work possible. She has been such an inspiration and encouragement during all of my academic endeavors. My parents, siblings, and grandparents who have been a constant presence in my life and have always energized me to work hard and have resolute tenacity. I love you all so much and am thankful you are in my life.

Acknowledgements

I am very thankful to several people who have helped me to create this work. My dissertation committee of Drs. Larry Miller, Hairong Qi, and Chester Ramsey for agreeing to spend their time and effort to be on my committee and for providing insight and recommendations which have been used to increase the usefulness of this dissertation. I would like to thank the physician and staff from the Gamma Knife center in Knoxville who over the past seven years have become close friends. Specifically I would like to thank Drs. David Hauge, Joel Ragland, Steven Sanders, and Richard Boyer for sharing his expertise regarding neuroscience and the methods of interventions for conditions of the cranium. I would also like to thank Drs. Joseph Meyer, Nilesch Patel, and John Anderson for continually educating me in the art of the utilization of ionizing radiation for the treatment of disease. I am thankful to Rita Latour for her skillful management of the Gamma Knife center and her essential support in the development of the work. Finally, I would like to thank Dr. Wesley Hines, my committee chair and faculty advisor, for being such a major part of this research. The idea for this work stemmed from a meeting between the two of us and his mentorship has been a key role in my professional life. My sincerest thanks to you all.

Abstract

Gamma Knife is a medical procedure that is used to treat several types of intracranial disease. The system utilizes gamma rays from Cobalt-60 radiation sources focused at an isocenter and a stereotactic frame system that serves as an immobilization device coordinate system. Treatment is performed by localizing the patient's disease with a medical imaging study and positioning the diseased area at the focused intersection of the beams. Patient treatment may require multiple treatment positions and varying beam sizes. The treatment position, time, and beam size is determined through a treatment planning process. Traditionally Gamma Knife treatment planning is performed manually by an expert planner. This process can be time consuming and arrival at an optimal plan may depend on the skill of the planner.

This work automates the treatment planning process with a multi-module optimization system. First, a kernel regression data mining module compares the treatment volume to a database of past treatment plans to create a set of initial plans. These plans seed a genetic algorithm optimizer that produces an optimized plan. The cost function for the optimization is a weighted average of several traditional metric for assessing stereotactic radiosurgery plan quality. A gradient descent optimizer is utilized to further refine the optimized treatment plan.

The developed system was applied to three Gamma Knife planning cases; a solitary metastasis, an acoustic schwannoma, and a pituitary adenoma. The system produced an average percent isodose coverage for the three plans of 94.5% and the average Paddick

Conformity index was 0.76 in an average time of 17.16 minutes for the three plans. The system was compared to an expert planner and an optimizer included with the Gamma Knife planning software. The developed system and expert planner performance was essentially equivalent (average percent isodose coverage 95.8%, average Paddick Conformity index 0.70, optimization time 20.52). The developed system performed much better than the Gamma Plan optimizer (average percent isodose coverage 85.8%, average Paddick Conformity index 0.71) however the Gamma Plan optimizer result was obtained quicker (optimization time about 1 minute). The developed system can be utilized for efficient high-quality Gamma Knife treatment planning.

Table of Contents

1	Introduction.....	1
1.1	Background and Research Need	4
1.2	Original Contributions	8
1.3	Dissertation Organization	10
2	Literature Survey	11
2.1	Leksell Gamma Knife Synopsis.....	11
2.1.1	Gamma Knife History	15
2.1.2	Indication & Contradictions for Gamma Knife Radiosurgery.....	15
2.1.3	Gamma Knife Device Overview.....	18
2.1.4	Gamma Knife Collimation System.....	18
2.1.5	Leksell Stereotactic Frame.....	22
2.1.6	Leksell Coordinate System Localization	22
2.1.7	Radiation Dose Distribution Calculation	24
2.1.8	Leksell Gamma Plan Treatment Planning	26
2.1.9	Gamma Knife Treatment Delivery	28
2.2	Existing Optimization of Gamma Knife Treatment Plans.....	30
2.2.1	Conformity Oriented Approaches.....	30
	Clinical Treatment Planning Optimization by Powell’s Method for Gamma Unit Treatment System	31
	Three-dimensional optimization of treatment planning for gamma unit treatment system	33
	Conformity Oriented Approached Summary.....	35

2.2.2	Simulated Annealing.....	36
	Treatment planning optimization by quasi-Newton and simulated annealing methods for gamma unit treatment system	36
	Optimization of Gamma Knife treatment planning via guided evolutionary simulated annealing.....	40
2.2.3	Mixed Integer Programming.....	43
	Optimization of Gamma Knife radio- surgery, in Discrete Mathematical Problems with Medical Applications.....	43
2.2.4	Non-linear programming	45
	Inverse treatment planning for Gamma Knife radiosurgery.....	46
	Clinical Implementation of an Automated Planning System for Gamma Knife Radiosurgery.....	48
2.2.5	Geometry Optimization Approaches	50
	Morphology-guided radiosurgery treatment planning and optimization for multiple isocenters	50
2.2.6	Data Mining Approaches	55
	Automated gamma knife radiosurgery treatment planning with image registration, data-mining, and Nelder-Mead simplex optimization	55
2.3	Empirical Modeling of Historical Plan Database	58
2.4	Genetic Algorithms.....	62
	A new genetic algorithm technique in optimization of permanent 125 Iodine prostate implants.....	66
2.5	Hybrid Optimization with Gradient Descent	67

3	Methodology	70
3.1	Importation of data from Gamma Plan to MatLab and CERR	73
3.2	Historical Treatment Plan Database.....	75
3.2.1	Kernel Regression Model for 2D Investigative Problem.....	76
3.2.2	Kernel Regression Model of Historical Plan Database.....	82
3.3	Initial Population by Geometric Optimization.....	90
3.4	Dose Calculation.....	92
3.5	Optimization Parameters & Fitness Function	97
3.6	Genetic Algorithm Optimization	102
3.6.1	Genetic Algorithm fitting of a One-Dimensional Test Problem.....	102
3.6.2	Genetic Algorithm Optimization of Gamma Knife Treatment Planning....	107
3.7	Gradient Descent Hybrid Optimization Module.....	113
4	Results.....	115
4.1	Solitary Metastasis Plan Optimization.....	115
4.1.1	Solitary Metastasis Initial Population Generation	116
4.1.2	Solitary Metastasis Genetic Algorithm Optimization.....	117
4.1.3	Solitary Metastasis Optimization Compared to Expert Planner	121
4.1.4	Solitary Metastasis Optimization Compared to Gamma Plan	122
4.2	Acoustic Schwannoma Plan Optimization.....	123
4.2.1	Acoustic Schwannoma Initial Population Generation	124
4.2.2	Acoustic Schwannoma Genetic Algorithm Optimization.....	126
4.2.3	Acoustic Schwannoma Optimization Compared to Expert Planner	128
4.2.4	Acoustic Schwannoma Optimization Compared to Gamma Plan	129

4.3	Pituitary Adenoma Plan Optimization	130
4.3.1	Pituitary Adenoma Initial Population Generation.....	132
4.3.2	Pituitary Adenoma Genetic Algorithm Optimization	133
4.3.3	Pituitary Adenoma Optimization Compared to Expert Planner	135
4.3.4	Pituitary Adenoma Optimization Compared to Gamma Plan.....	137
4.4	Summary of Optimization Results.....	138
5	Conclusions and Suggestions for Future Work.....	140
5.1	Summary of original contributions	140
5.2	Future Work	144
	Reference	146
	Appendices.....	155
	Appendix A: MatLab Code.....	156
	Appendix B: Figures and Tables.....	348
	Appendix C: Institutional Review Board Exemption Documentation	401
	Vita	410

List of Table

Table 1 Investigative Database Parameters	78
Table 2 Test shape 1 similarity for each database shape	80
Table 3 Test shape 2 similarity for each database shape	82
Table 4 Historical Treatment Volume Features.....	87
Table 5 Percent Isodose Volume cm3 for each collimator by coverage parameter.....	91
Table 6 Genetic Algorithm Optimization of Test Problem Results.....	107
Table 7 Solitary Metastasis Treatment Plan Parameters.....	120
Table 8 Acoustic Schwannoma Treatment Plan Parameters	128
Table 9 Pituitary Adenoma Treatment Plan Parameters.....	135
Table 10 Test Case Summary	139
Appendix Table 11 Historical Database Treatment Volume Table.....	353
Appendix Table 12 Historical Database Treatment Plan Table.....	357

List of Figures

Figure 1 Typical Gamma Knife treatment day process	5
Figure 2 Treatment MRI with structures designated	6
Figure 3 Gamma Knife Optimization Process	8
Figure 4 Gamma Knife Timeline.....	16
Figure 5 Leksell Gamma Knife C	19
Figure 6 Gamma Knife hemispherical shielding of Cobalt-60 sources	20
Figure 7 Gamma Knife single source channel collimation system.....	21
Figure 8 Leksell Stereotactic Frame	23
Figure 9 Leksell MRI Fiducial Coordinate Box	23
Figure 10 Fiducial System Imposed on MRI Slice	24
Figure 11 Gamma Plan Calculation Geometry	27
Figure 12 Gamma Knife Treatment Planning 1.....	29
Figure 13 Gamma Knife Treatment Planning 2.....	29
Figure 14 Skeletonization Example	51
Figure 15 Kernel Regression (a-inferential, b-heteroassociative, c-autoassociative) [66]	60
Figure 16 Kernel Regression [66].....	61
Figure 17 Gaussian Kernel Function with varying bandwidth h [66]	62
Figure 18 Genetic Algorithm Flowchart.....	64
Figure 19 Gradient Descent	68
Figure 20 Treatment Plan Optimization Overview.....	71
Figure 21 CERR importation of Gamma Knife treatment plan.....	75
Figure 22 Investigation Database Shape 1	77

Figure 23 Investigation Problem Input Shape 1	79
Figure 24 Input Test Shape 1 Overlaid on Database Shape 1.....	80
Figure 25 Investigation Problem Input Shape 1	81
Figure 26 Input Test Shape 2 Overlaid on Database Shape 10.....	83
Figure 27 Principal Components	83
Figure 28 Historical Plan Example Treatment Volume.....	85
Figure 29 Historical Plan Database Structure	86
Figure 30 Isodose Distribution Rotation about Y axis.....	90
Figure 31 Geometric Optimization	92
Figure 32 Gamma Knife radiation distribution near edge of phantom.....	94
Figure 33 Gamma Plan versus Rapid Dose Calculation Comparison X Plane.....	95
Figure 34 Gamma Plan versus Rapid Dose Gamma Distribution Transverse Plane.....	97
Figure 35 GA Test Problem Data	104
Figure 36 Genetic Algorithm Optimization of Test Problem	106
Figure 37 TOGA Genetic Algorithm Optimization GUI.....	108
Figure 38 Solitary Metastasis Target Volume	116
Figure 39 Solitary Metastasis Optimization	119
Figure 40 Solitary Metastasis Optimization Isodose Coverage.....	119
Figure 41 Solitary Metastasis Optimization Plan versus Expert Planner	121
Figure 42 Solitary Metastasis Optimization Plan versus Gamma Plan Optimizer	123
Figure 43 Acoustic Schwannoma Target Volume	124
Figure 44 Acoustic Schwannoma Optimization	127
Figure 45 Acoustic Schwannoma Optimization Isodose Coverage.....	127

Figure 46 Acoustic Schwannoma Optimization Plan vs. Expert Planner.....	129
Figure 47 Acoustic Tumor Optimized Plan vs. Gamma Plan Optimizer	130
Figure 48 Pituitary Adenoma Target Volume	131
Figure 49 Pituitary Adenoma Optimization.....	134
Figure 50 Pituitary Adenoma Optimization Isodose Coverage	134
Figure 51 Dose Volume Histogram of Pituitary Adenoma Treatment Plan.....	136
Figure 52 Pituitary Adenoma Optimization Plan vs. Expert Planner	136
Figure 53 Pituitary Adenoma Optimized Plan vs. Gamma Plan Optimizer	138
Appendix Figure 54 Investigation Database Shape 2	348
Appendix Figure 55 Investigation Database Shape 3	348
Appendix Figure 56 Investigation Database Shape 4	349
Appendix Figure 57 Investigation Database Shape 5	349
Appendix Figure 58 Investigation Database Shape 6	350
Appendix Figure 59 Investigation Database Shape 7	350
Appendix Figure 60 Investigation Database Shape 8	351
Appendix Figure 61 Investigation Database Shape 9	351
Appendix Figure 62 Investigation Database Shape 10	352
Appendix Figure 63 Histogram of Historical Plan Database Volumes	396
Appendix Figure 64 Histogram of Historical Plan Principal Component 1 Length.....	396
Appendix Figure 65 Histogram of Historical Plan Principal Component 2 Length.....	397
Appendix Figure 66 Histogram of Historical Plan Principal Component 3 Length.....	397
Appendix Figure 67 Histogram of Irregularity Parameter.....	398
Appendix Figure 68 Histogram of Historical Plan Treatment Isocenter Weight	398

Appendix Figure 69 Gamma Plan vs. Rapid Dose Calc Comparison Y Plane.....	399
Appendix Figure 70 Gamma Plan vs. Rapid Dose Calc Comparison Z Plane	399
Appendix Figure 71 Gamma Plan vs Rapid Radiation Gamma Coronal Plane	400
Appendix Figure 72 Gamma Plan vs Rapid Radiation Gamma Sagital Plane	400

Acronyms

AVM - Arteriovenous Malformation

BRGK – Baptist Regional Gamma Knife

CERR - Computational Environment for Radiotherapy Research

COM – Center of Mass

CT – Computed Tomography

DICOM-RT - Digital Imaging and Communications in Medicine Radiotherapy

GA – Genetic Algorithm

IV – Intravenous

MatLab – Matrix Laboratory

MRI – Magnetic Resonance Imaging

PET – Positron Emission Tomography

ROI – Region of Interest

RTOG – Radiation Therapy Oncology Group

TOGA – Treatment planning Optimization via Genetic Algorithm

1 Introduction

Radiation oncology is a discipline of medicine concerned with the treatment of cancer and other disease by means of the application of therapeutic ionizing radiation [1]. The beginnings of the field can be traced to the 19th century when pioneer physicist Wilhelm Rontgen discovered x-rays. Shortly after Rontgen's discovery, Becquerel realized the diagnostic utility of x-rays. Radiation oncology was further expanded by other scientists such as Marie and Pierre Curie who discovered radium and polonium, as these radio-isotopes would later be used for the treatment of disease. In the mid-1900s, medical linear accelerators were introduced making radiation treatments more widely available and alleviating some of the safety concerns of radio-isotopes therapy. Medical Linear Accelerators produce high energy x-rays (greater than 1 mega electron volt maximum energy) and currently are the most utilized method of delivering radiation therapy [2].

Today, radiation therapy has become an integral part of many cancer patients' treatment regimens. In fact, about half of all cancer patients receive some form of radiation therapy. Radiation therapy may be used alone or in combination with other cancer treatments, such as chemotherapy (chemical therapy) or surgery [3]. Malignant and benign tumors, in addition to several functional disorders, and arterial venous malformations (AVM) are examples of disease treated with radiation [4].

The aim of the radiation therapy may be cure or palliation of disease depending on the pathology and the extent of the disease. The treatment may be delivered from an external source (teletherapy), from an internal source placed inside the body (brachytherapy), or by

an injected or ingested liquid that travel throughout the body (systemic therapy).

Teletherapy is the most common of these.

When a patient is irradiated, ionizing radiation damages cancer cells ultimately leading to their death and in turn the control of the disease. The method by which radiation causes cell death is complex and currently not completely understood. The process is outside the scope of this work and will only be discussed briefly here. Please refer to Hall [5] for a more detailed discussion of radiobiology. Radiation treats the diseases by damaging the DNA of the tumor cells prohibiting the cells from further dividing or prohibiting future cell generations from division. The damage is in the form of breaks in the DNA strand. The damage occurs when the radiation interacts and creates free radicals that interact with the DNA through energy deposition resulting in breaks in the DNA strands. These breaks may prohibit the cancer cells from further dividing, thus controlling the disease.

Unfortunately, radiation will damage the normal cells as well as the diseased cells; however, the normal cells have mechanisms for the **repair** of sublethal damage over the course of the treatment. Normal tissue will also **repopulate** the number of cells over the course of the treatment. Cancer cells will **redistribute** over the cell cycle during the treatment resulting in the cells being more radiosensitive. Finally, over the course of the treatment the tumor may shrink causing the **re-oxygenation** of the hypoxic center of the tumor. Oxygen acts as a radio-sensitizer by serving as a free radical reservoir. These four principals, the four “Rs” of radiobiology, describe the principals that are used to treat cancer with radiation.

Ideally, radiation therapy is applied directly to the tumor, while limiting exposure to the normal tissue in the body. This is not possible with systemic therapy as the radio-isotope is distributed throughout the body. Brachytherapy can easily target only the disease because the radiation source is contained within the volume to be treated. However, these treatments are invasive and not an option for many types of disease. Teletherapy delivered from an external source is able to selectively target disease by using radiation beams from multiple angles that are shaped to conform to the treatment area. The selection of treatment angles and radiation beam shape for teletherapy is a process known as treatment planning [6].

Treatment planning consists of two steps. First the patient's disease is localized in their body by an imaging simulation. The imaging simulation may be a computed tomography (CT) scan, a magnetic resonance imaging (MRI) scan, a positron emission tomography (PET) scan or a simple plane film x-ray. After simulation, several machine parameters are selected that allow the radiation to target the disease while limiting the dose of radiation to nearby non-diseased organs in the body. Multiple angles are commonly utilized because the radiation dose will be additive at the intersection of the beam (the diseased area) and the integral radiation dose not delivered to the diseased area will be spread elsewhere over the body. In addition to the tumor, the radiation treatment area may include draining lymph nodes where the disease may have spread. Physicians and physicist utilize the addition of the radiation at the beams intersection to selectively target the disease while sparing the normal tissue that surrounds. As the process has become more refined with experience, physicians have been able to deliver higher

uniform radiation doses to the disease. Radiation therapy can ultimately be described as an optimization with the goal being the highest uniform dose of radiation possible to the disease and the lowest dose of radiation possible to the rest of the body.

Today, the treatment of disease with radiation has become a standard for many types of medical disorders due to its past success and the advent of advanced imaging systems.

CT, MRI, and PET imaging systems allow for the early diagnosis and accurate localization of disease. As the disease is better localized, every attempt is made to treat the diseased area with the highest radiation dose possible while limiting the dose to normal tissue, especially normal tissue that may be more sensitive to radiation damage.

This is achieved by optimizing various parameters during the treatment planning such as position of the beam, size of the beams, and time the beam is used to treat. Computers have been introduced to perform many of these optimizations to improve the quality of radiation treatments. The purpose of this work is to develop an optimization routine to optimize a specific type of radiation therapy called Gamma Knife radiosurgery. Gamma Knife is a type of external beam radiation therapy that utilizes Cobalt-60 isotope sources to treat diseases of the brain.

1.1 Background and Research Need

Gamma Knife is a medical procedure that is used to treat several types of intracranial disease [4]. The system utilizes the gamma rays from multiple Cobalt-60 radiation sources focused at a precise point and a stereotactic frame system that acts as an immobilization device and the foundation for the system's coordinate system. The flow chart in figure 1 illustrates the steps involved in a typical Gamma Knife treatment [7].

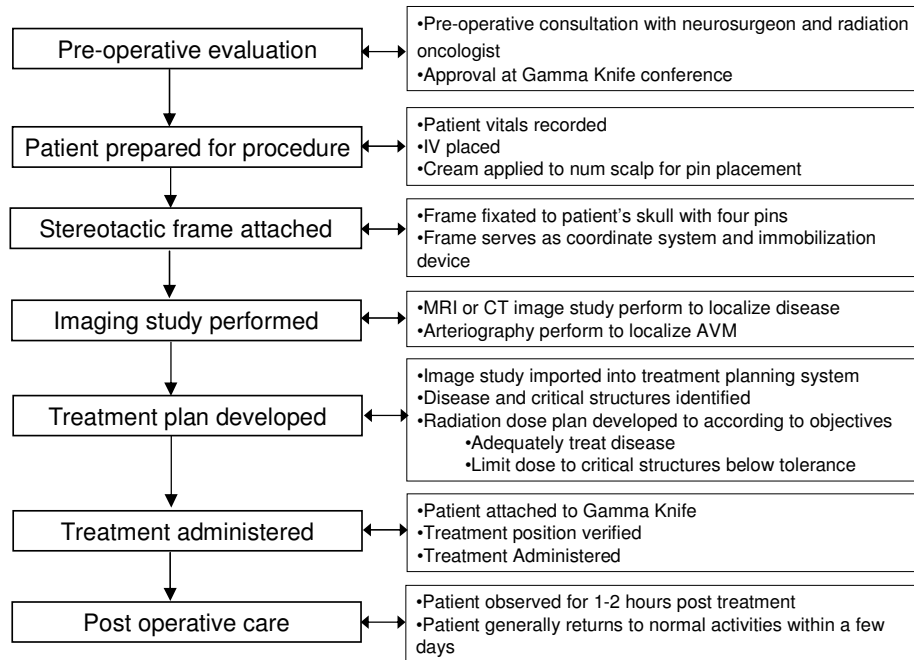


Figure 1 Typical Gamma Knife treatment day process

Treatment begins with a neurosurgeon attaching a rigid stereotactic frame to the patient's skull using four screws. A medical image study is then performed and a physician delineates a treatment area and any radio-sensitive structures in close proximity to the treatment area. Figure 2 displays several slices from a MRI image study with the treatment area, a pituitary tumor, outlined in red, the left optic nerve outlined in blue, the right optic nerve outlined in green, and the optic chiasm outlined in magenta. Note the close proximity of these radiosensitive structures, generally about 1 or 2 mm separation, to the pituitary tumor. Pituitary tumors are a good example of the need for a conformal radiation treatment plan delivered accurately. Failure to produce a conformal plan, or accurately deliver the treatment could potentially blind the patient as the optic nerves and chiasm are radiosensitive structures.

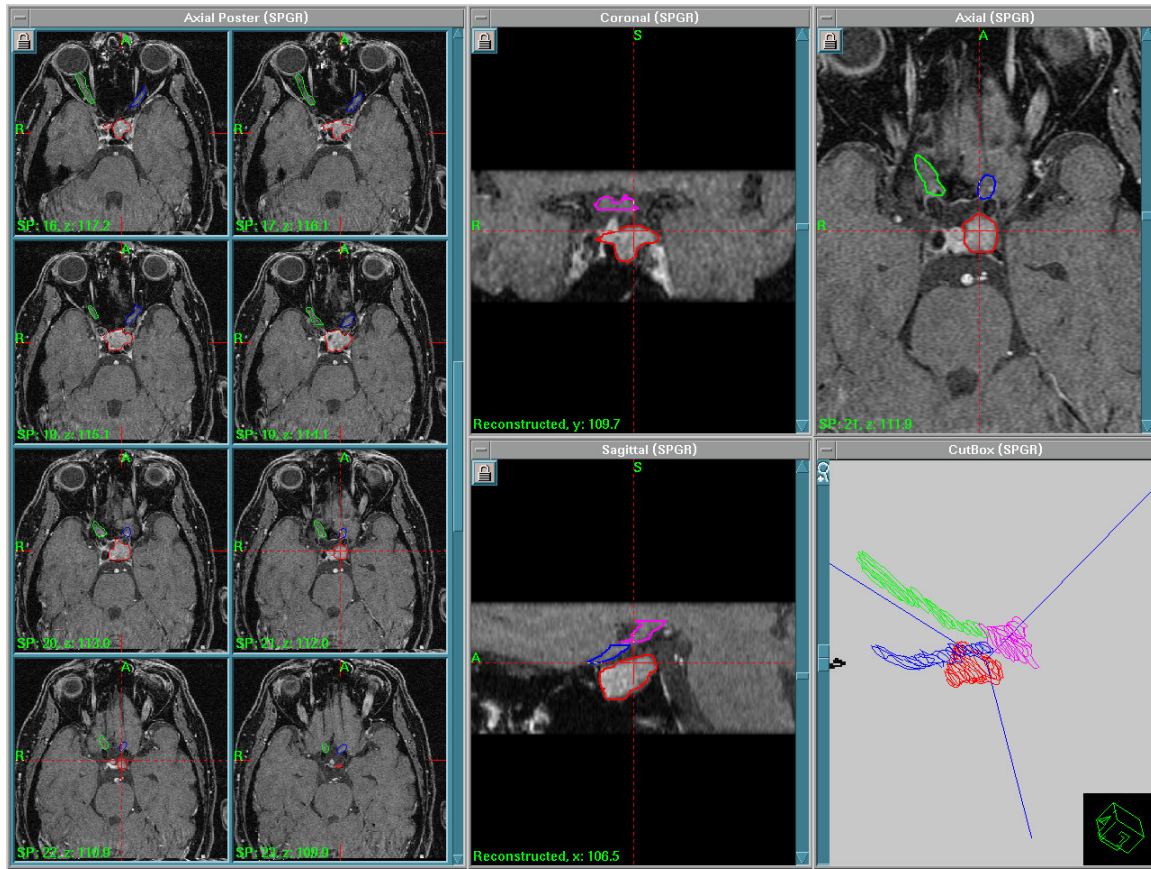


Figure 2 Treatment MRI with structures designated

During treatment, the diseased area is placed at the focus of the radiation sources for a calculated amount of time. The calculated time is determined through a treatment planning process where radiation isocenters, sometimes called shots, of varying size are placed at various locations throughout the treatment area to produce a radiation dose distribution that conforms to the treatment area, while sparing normal tissue and critical structures in the brain [8]. The process is currently performed by an expert planner who manually changes the location, size, and time of the radiation shots to produce an optimal plan. The treatment planning process can be time consuming, sometimes taking as long as an hour, and is not guaranteed to produce an optimal treatment plan. The quality of

the treatment plan depends on the proficiency of the planner. A less experienced planner may not produce an optimal plan or take a longer time to produce a suitable treatment plan. It is desirable to produce a quality plan quickly as the patient is waiting to be treated with the frame attached to their skull. Several factors influence the time required for treatment planning including the size of the treatment volume, proximity of the treatment volume to radio-sensitive structures, and the degree of irregularity of the shape of the treatment volume. Long planning times are obviously undesirable and in the interest of patient comfort and quality of patient care, it is desirable to produce a quality radiation treatment plan in as short amount of time as possible. A computer optimization system should be ideally suited to achieve this objective.

The purpose of this work is to automate the treatment planning process with a multi-module optimization system that will produce a quality radiation treatment plan quickly. First, a kernel regression data mining module compares the treatment area to a database of past treatment plans and creates a set of initial similar treatment plans. A geometrical optimizer is then utilized to intelligently match the data mined treatment plans to the current treatment volume. These initial plans will seed a genetic algorithm optimization module that creates an optimized treatment plan. Additionally, a gradient descent optimizer may be utilized to further refine the genetic algorithm optimized plan. The developed system is capable of decreasing planning times while producing conformal treatment plan comparable to that of an expert planner. Figure 3 is a flow chart that illustrates the complete optimization process.

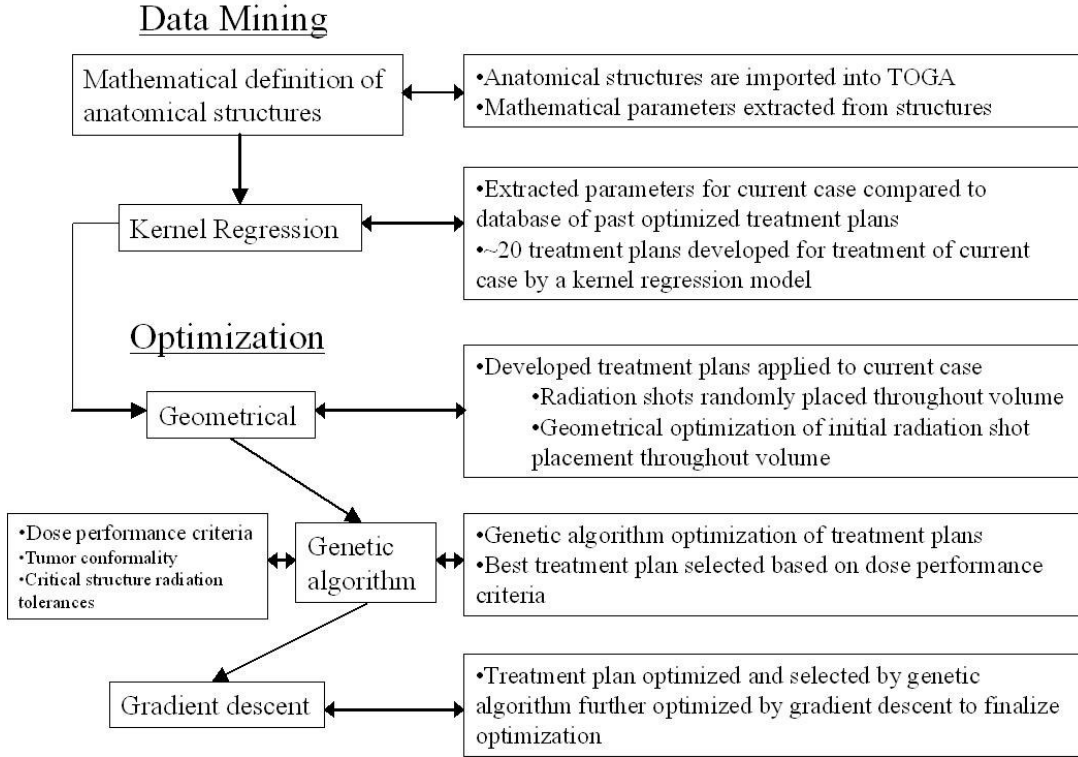


Figure 3 Gamma Knife Optimization Process

1.2 Original Contributions

Several authors have utilized different techniques to develop an inverse algorithm to automate the Gamma Knife treatment planning process. These works will be discussed within the literature survey in chapter 2. This work will automate Gamma Knife treatment planning with a method not yet attempted. The following is a summary of the contributions of this dissertation:

1. Development of an initialization module that generates several new treatment plans that are appropriate to treat the physician designated area. This is accomplished by extracting several features from the current target volume.

These features are compared to a database of past treatment volume features to

locate several historical treatment volumes that are similar to the current case. The module then extracted a set of treatment plans that correspond to the similar historical treatment volumes and should be appropriate for the current treatment. The result of this module seeds a genetic algorithm optimization module.

2. Development of a genetic algorithm optimization module to optimize the seeded treatment plans to produce a single treatment plan that conforms to the designated treatment volume. Genetic algorithms are a search technique used in physics and engineering applications to find approximate solutions to optimization and search problems. This system optimizes the created plans to produce an optimized treatment plan.
3. Development of a rapid radiation dose distribution for the genetic algorithm optimization. The genetic algorithm optimization requires the dose distribution from approximately 20-30 treatment plans to be calculated at each optimization iteration. To quickly produce an optimized plan it is necessary to rapidly perform the radiation dose transport calculation.
4. Development of a gradient descent optimization module that refines the result of the genetic algorithm optimization to ensure that the system as a whole produces the optimal treatment plan.
5. Integration of all developed modules into a single computer program that quickly and efficiently optimizes a Gamma Knife treatment plan. The integrated code has been termed TOGA, **T**reatment planning **O**ptimization via **G**enetic **A**lgorithms.

1.3 Dissertation Organization

The dissertation is organized within five chapters followed by the references, appendices, and vita. An introduction is presented in Chapter 1 that includes background, original contributions and document organization. Chapter 2 is a literature review that gives a detailed survey of the entire Gamma Knife treatment process. This description is followed by a review of other optimization methods for Gamma Knife radiosurgery treatment planning. Finally, a description of genetic algorithms and kernel regression data mining, the techniques used in this work to optimize Gamma Knife treatment planning is presented. Chapter 3, the methodology, gives an in depth description of the techniques that were created and the algorithms that were used to accomplish the research goals. Chapter 4 presents results for the given methods. Finally, the conclusion contained in Chapter 5 summarizes the research that was undertaken, discusses the original contributions and the research that was done to accomplish the given tasks, and gives recommendations for future work.

2 Literature Survey

The following literature survey first presents a detailed description of the entire Gamma Knife treatment. This description is followed by a review of other optimization methods for Gamma Knife radiosurgery treatment planning. Finally, a description of genetic algorithms and kernel regression data mining, the techniques utilized in this work to optimize Gamma Knife treatment planning is presented.

2.1 Leksell Gamma Knife Synopsis

The Leksell Gamma Knife is a neurosurgical device that is used to treat benign and malignant intracranial conditions [4, 7, 9]. Contrary to the name, the Gamma Knife is not an actually knife, but rather a sophisticated system that utilizes Cobalt-60 radiation sources to irradiate targeted areas of the brain to treat disease. The device focuses gamma radiation at a target point in the patient's brain. The patient wears a specialized frame that is surgically fixed to their skull so that the brain tumor remains stationary at the target point of the gamma rays. A lethal dose of radiation is thereby sent through the tumor in one treatment session, while all surrounding brain tissues receive less than a lethal dose. The original device was developed by Swedish neurosurgeon Lars Leksell in 1967 at the Karolinska Institute in Sweden. The Leksell Gamma Knife has undergone numerous refinements, additions, and modifications since 1967 and refinements are still ongoing today. The work presented here is focused on the Leksell Model C Gamma Knife.

Patient treatment is a multi-step process. A head frame is fixated to the patient's skull at four pin sites. An imaging study is then performed with a fiducial box attached to the frame. The fiducial box is used to define the patient's anatomy in the Leksell coordinate system. Once patient anatomy is defined, a treatment site is delineated and a treatment plan developed. Planning can be time consuming as multiple parameters must be considered to develop a treatment plan that is optimal for treatment of the patient's tumor. Once a plan is complete, the patient is attached to the Gamma Knife using the frame according to the treatment plan and the patient is treated.

Gamma Knife utilizes Cobalt-60 radiation beams that are focused at a central point known as the isocenter of the system. The intersection of the radiation beams at the isocenter results in a high radiation dose at the isocenter and a relatively small radiation dose in the area surrounding the isocenter. Patient treatment is performed by placing the treatment site at the isocenter. The patient treatments can be tailored to the particular treatment site by varying several parameters. First, the radiation beams are focused using four helmets of various size (4mm, 8mm, 14mm, & 18mm), resulting in different size radiation distribution at the isocenter. The different sized helmets can be used to treat treatment sites of varying size. Second, the patient can be placed in the Gamma Knife at "gamma angles" of 72, 90, 110, and 125 degrees. The gamma angle is the angle the patient's frame to the Z axis of the Gamma Knife. Third, the X, Y, & Z coordinates that the patient is placed in the Gamma Knife can be set to optimize the coverage of the radiation distribution to the treatment site. Fourth, individual radiation beams can be "blocked" using steel plugs to protect critical structures in the brain. Finally, patients can

be treated with multiple isocenters at different coordinates, helmet sizes, and angles. The multiple shots can be given varying weights (treatment time) resulting in complex irregularly shaped radiation distributions that mimic irregularly shaped treatment sites. Treatment planning is thus a combination of several treatment parameters that must be optimized to develop the most conformal treatment plan for the patient.

The Gamma Knife is a dedicated stereotactic radiosurgery device. Several professional organizations have developed definition for stereotactic radiosurgery. In 1993 the American Society of Therapeutic Radiation Oncology and the American Association of Neurosurgery issued a joint statement that defined stereotactic radiosurgery as “a distinct discipline that utilizes externally generated ionizing radiation in certain cases to inactivate or eradicate defined targets in the head and spine without the need to make an incision. The target is defined by high-resolution stereotactic imaging [10].” The American Association of Physicist in Medicine Task Group 54 defined stereotactic radiosurgery as "the use of multiple narrow radiation beams in a non-coplanar arc to deliver a single fraction of high dose radiation therapy for the treatment of intracranial lesions using a stereotactic apparatus such as a stereotactic frame [11]. The International Radiosurgery Society defines radiosurgery as the “Use of a number of precisely aimed, highly focused beams of ionizing radiation to target a specific area [12].” Radiosurgery is a non-invasive procedure that utilizes precise dosage of radiation to treat intracranial tumors and other lesions that may be inaccessible or inadequate for open surgery. Physicians perform Gamma Knife radiosurgery to treat tumor and other types of disease in areas of the brain that may be inaccessible for open surgery, or in patients who are not

healthy enough for surgery, or to avoid an open operation. The Gamma Knife is used only for cranial stereotactic radiosurgery and has the highest accuracy of all available radiosurgical devices [13].

The Gamma Knife has an intensive treatment track record with over 350,000 treatments performed [7, 14]. There are 220 Gamma Knife centers on 6 continents, 100 in the United States and two in Tennessee (Knoxville and Memphis). The device has been in continuous use in Europe since 1969 and in the United States since 1986. There are more than 2,500 peer-reviewed articles supporting safety and efficacy of Gamma Knife radiosurgery.

The Gamma Knife has a system accuracy of 0.3 mm [9]. The system accuracy describes the ability of the device to accurately place a point within the frame at the center of the Gamma Knife treatment beam. The system employs a rigid stereotactic frame fixation and several hundred stationary Cobalt 60 sources to maximize treatment accuracy and patient safety. The system has a manual treatment mode known as trunnions and an Automatic Positioning System (APS) that utilizes robotic technology to move patient sequentially to each radiation treatment shot. Gamma Knife is a single session treatment modality. In the case of extremely large treatment area, a portion of the disease may be treated in one session and the remainder at a later time to limit the amount of radiation delivered in one fraction to the patient.

2.1.1 Gamma Knife History

Open brain surgery in the first half of the twentieth century was described as consisting mostly of "plenty of blood, sweat, and tears" by Gamma Knife inventor and pioneer Swedish neurosurgeon Lars Leksell [7, 14]. In the late 1940's, Dr. Leksell began to look for less invasive ways to treat intracranial disorders and collaborated with radiation physicists at the Karolinska Institute in Stockholm. By the late 1960's, the first generation Model U Gamma Knife was being used to treat neurological disorders such as trigeminal neuralgia, Parkinson's disease and arteriovenous malformations.

Since the development of the original Model U Gamma Knife, this technology has been continuously researched and upgraded. The current technology is in its fifth generation with this research based on a fourth generation Model C Gamma Knife, installed at Baptist Regional Gamma Knife Center in Knoxville Tennessee. Figure 4 is a timeline displaying some high points in the history of the Gamma Knife.

2.1.2 Indication & Contradictions for Gamma Knife Radiosurgery

The last two decades have seen a rapid expansion of the indications for Gamma Knife treatment to include virtually all types of intracranial neoplasms, trigeminal neuralgia, arteriovenous malformations (AVMs) and selected cases of certain functional disorders such as Parkinson's disease and essential tremor [7, 14]. Current research is focusing on Gamma Knife treatment for medically intractable epilepsy, glaucoma, and certain psychiatric disorders such as obsessive compulsive disorder.

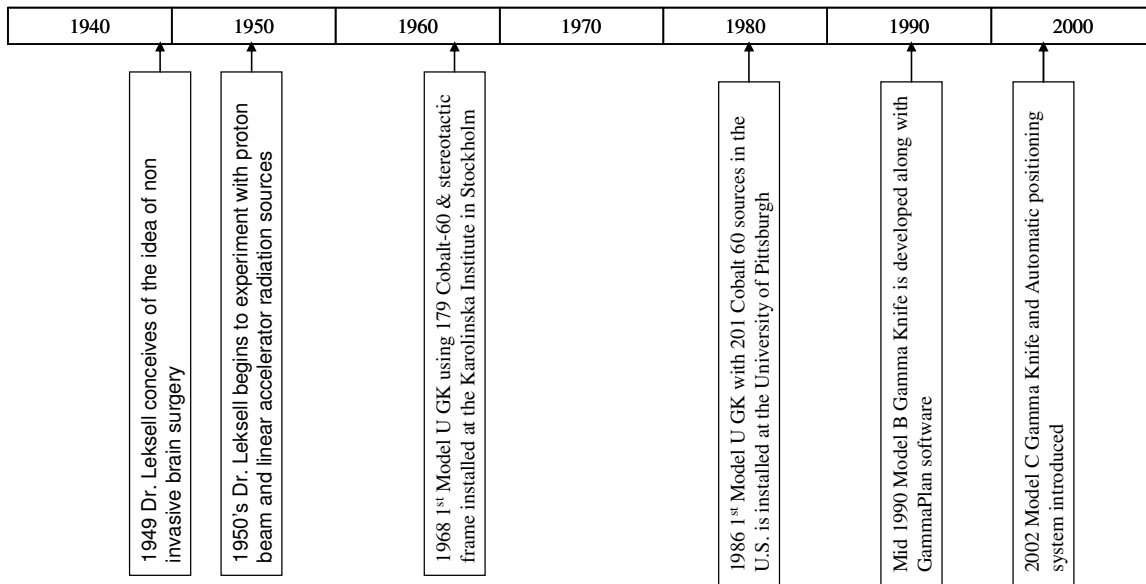


Figure 4 Gamma Knife Timeline

Specific indications for Gamma Knife Radiosurgery include:

- Tumors: Tumors up to 3-4 cm in size can be treated with Gamma Knife radiosurgery in a single treatment session. Certain larger tumors and AVMs can now be treated in multiple staged single session treatments spaced several months apart. Gamma Knife is also performed as part of a multidisciplinary treatment plan for certain types of tumors. Many tumors exceed 3-4 cm at the time of initial diagnosis. These tumors must frequently be treated with open surgery to debulk in size. Large portions of these tumors can be removed while tumor areas adjacent to critical structures can be treated later with Gamma Knife radiosurgery with much less neurological risk. Standard fractionated radiotherapy and chemotherapy are also frequently used in conjunction with Gamma Knife. Because of the precise conformal treatment possible with Gamma Knife, patients

who develop multiple metastatic lesions may undergo multiple Gamma Knife treatments throughout the course of their illness.

- Acoustic schwannoma
- Meningioma
- Pituitary tumors
- Primary glial brain tumors
- Trigeminal neuralgia
- Arteriovenous malformations (AVMs)

There are few contraindications to Gamma Knife radiosurgery. The patient must be cooperative and be able to undergo MRI or CT imaging. Patients with multiple co-morbidities (such as chronic obstructive pulmonary disease, emphysema and congestive heart failure) can undergo Gamma Knife treatment with much less risk than open surgery.

Gamma Knife treatment can be performed safely without interruption of any anticoagulation therapies the patient may be on. Oxygen and intravenous fluid administration can be continued during Gamma Knife treatment. ECG monitoring and routine pulse oximetry can be performed as well. Gamma Knife patients require very little specialized care following the procedure.

In general, patients with end stage cancer and other terminal illnesses are considered for treatment if their Karnofsky performance status is 60 or greater and have a life expectancy of six months or longer. Exceptions to this may be made on a case by case basis.

2.1.3 Gamma Knife Device Overview

The Leksell Gamma Knife, marketed by Elekta, is an efficient non-invasive radiosurgery system that uses Cobalt-60 radiation sources to treat intracranial tumors and lesions [9, 15]. Cobalt-60 has a physical half life of 5.27 years. The dose rate of the machine at loading is greater than 3 Gray per minute. The physical dimensions of the machine are 4.64 meter in length, 2.00 meters in length, and 1.935 meter in height. The total weight of the machine is approximately 20,000 kg. The devices consist of the spherical shielded housing where the radiation sources are held, a moveable couch that the patient lies on during treatment, and four collimator helmets that shape the radiation beam. The patient is attached to the couch with an immobilization device at a planned position and with a planned helmet to treat their disease. During treatment two shielding doors open, the couch moves into the shielded housing and docks with the inside of the housing while the patient receives the treatment. There are no moving parts during treatment therefore safety, stability, and accuracy is inherent in the design. At the end of treatment the couch is removed from the shielded housing and the shielding doors close. Figure 5 is a picture of a Gamma Knife model C, the same Gamma Knife in place at the Baptist Regional Gamma Knife Center in Knoxville Tennessee.

2.1.4 Gamma Knife Collimation System

The Cobalt-60 radiation sources are arranged in a hemispherical pattern, embedded in the Gamma Knife shielding illustrated in figure 6 [9, 15]. The gamma radiation that emits from the Cobalt-60 sources is allowed to pass through a two component beam shaping system. This system directs and shapes the beam producing the radiation shot. After passing through the collimator system, the radiation from the sources all coincide at a

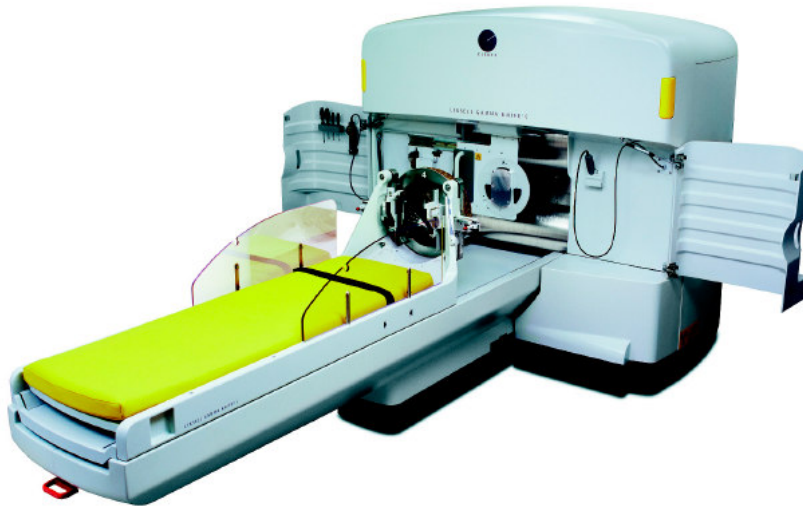


Figure 5 Leksell Gamma Knife C

fixed point called the isocenter of the machine. The radiation beam from each individual source is radio-biologically weak, however when the sources meet and add together at the isocenter they produce an area of high radiation. This area of high radiation constitutes the radiation shot, and it is this area that is used to treat disease. The collimator system that defines the shot consists of several hundred beam channels each with a primary and secondary collimator. The primary collimator is milled to progressively increase diameter away from the source as illustrated in figure 7. This progressive increase in diameter prevents the production of a transmission penumbra. Because the gamma radiation is emitted from essentially a point source, the beam will spread as it gets further from the source. The collimator is milled to match the spread of the beam. This results

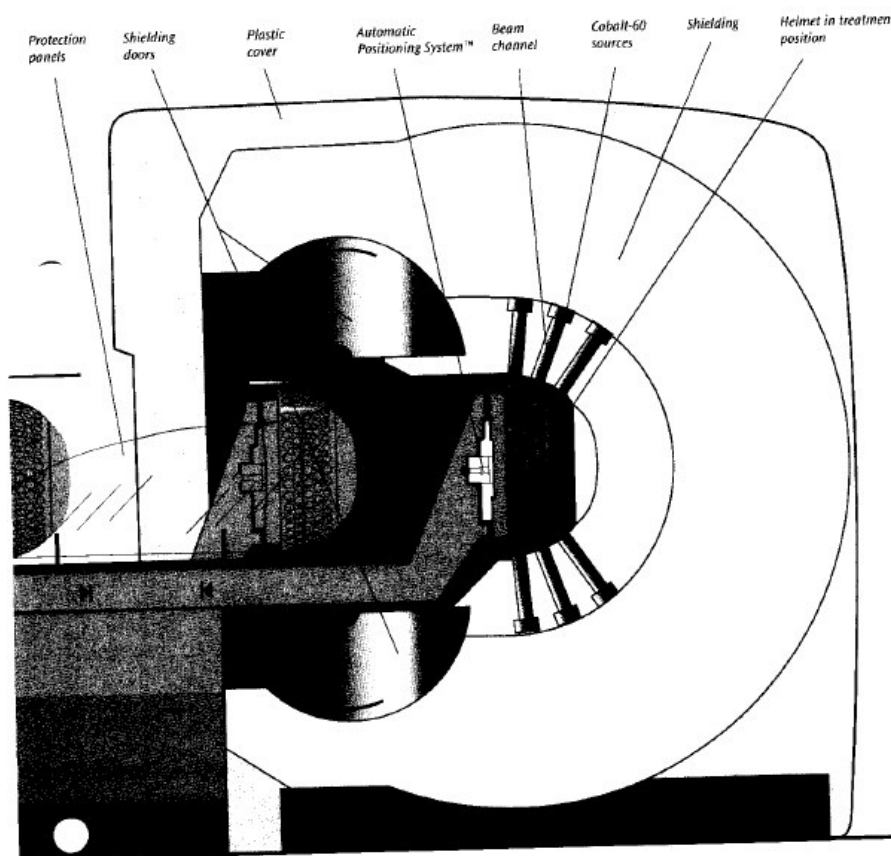


Figure 6 Gamma Knife hemispherical shielding of Cobalt-60 sources

in the decrease in dose from the high dose area of the radiation shot to the low dose outside of the shot as quick as possible. The primary collimator is stationary and not interchangeable. The radiation beam emitted from the Cobalt-60 sources is initially shaped by the primary collimator. The secondary collimator, commonly referred to as the helmet, is not stationary and is interchangeable. The secondary collimator is also milled to match and spread of the beam with a 4, 8, 14, or 18 mm in radio-physical dimension depending on the selected collimator. Radio-physical dimension means the size of the useful or treatment radiation beam. This is defined as 80% of the maximum dose point, or the 80% isodose of the helmet radiation dose distribution. The size of the

radiation dose distribution is determined by the selected helmet. Different helmets are selected depending on the size and shape of the diseased area to be treated. The helmet is locked into position with primary collimator by a spring loaded bearing. The radiation distribution can be further shaped by “plugging” individual beam channels of the helmet with stainless steel plugs that prohibit the beam from passing through the beam channel. The plugs have approximately 1% beam transmission. Plugging is useful when a beam channel may pass through a particularly radio-sensitive area of the brain.

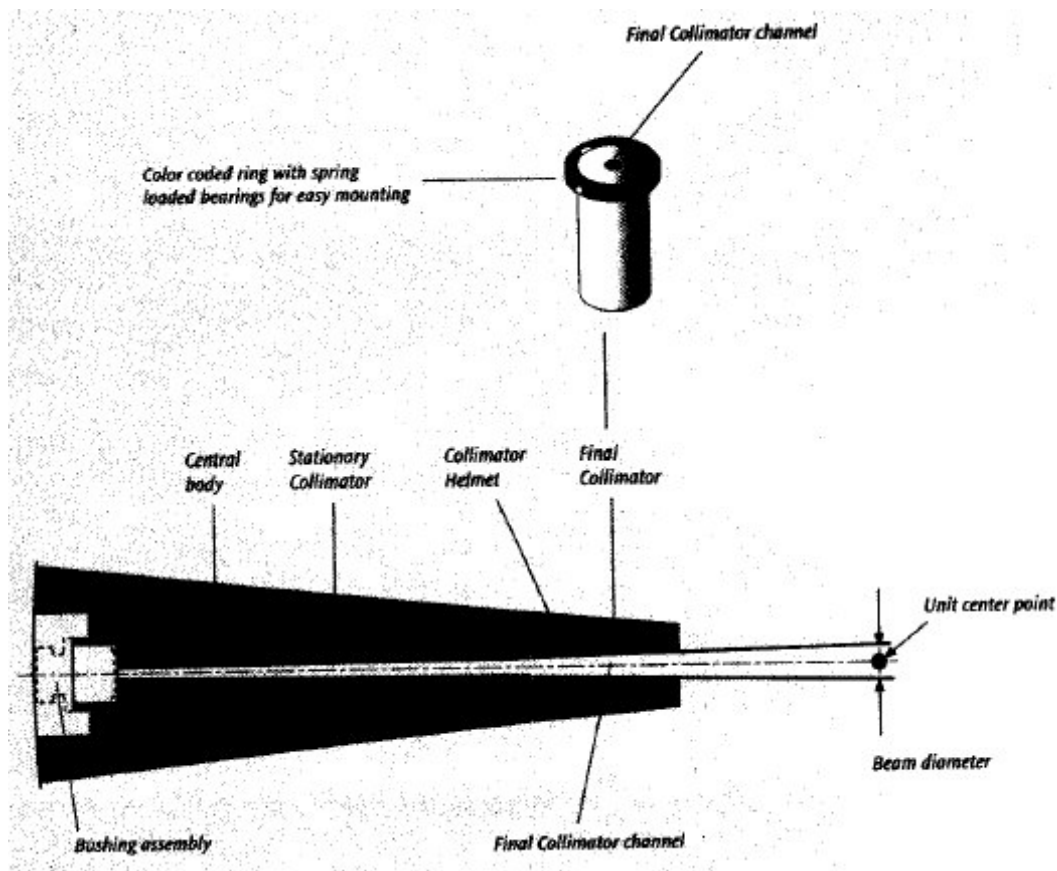


Figure 7 Gamma Knife single source channel collimation system

2.1.5 Leksell Stereotactic Frame

The Leksell stereotactic coordinate frame is the foundation of the Gamma Knife stereotactic system. The frame is used to immobilize the patient and as the basis for the coordinate system for treatment. The frame is made of titanium and aluminum alloy and is both CT and MR compatible. The frame is fixated to the patient's skull with 4 quick fixation screws by a neurosurgeon. A local anesthetic is applied to the pin site to reduce patient discomfort. The screws penetrate the skull to prevent slippage of the frame from the patient during the treatment. The frame is rigid and is used to define the Leksell coordinate system that is used during the Gamma Knife treatment. The x coordinate of the Leksell coordinate system is in the coronal plane of the patient increasing from patient left to right. The y coordinate is in the sagittal plane increasing from posterior to anterior. The z coordinate is in the axial plane increasing from superior to inferior. Figure 8 displays a computer rendering of the frame attached to a patient. This coordinate system in the y direction can be read on the side of the frame that is visible.

2.1.6 Leksell Coordinate System Localization

Treatment planning is performed with a commercial integrated computer treatment planning system known as Leksell Gamma Plan specifically designed for Gamma Knife stereotactic radiosurgery. The system utilizes MR, CT, and or arteriogram images to localize the disease. The images are imported in the computer, and registered in the Leksell coordinate space. This is accomplished using a fiducial box attached to the stereotactic frame. Figure 9 display the MR fiducial box. The fiducial boxes are plastic and attach to the stereotactic frame. The CT box contains aluminum wires in a known arrangement. The MR box contains cupric sulfate solution in channels in a known

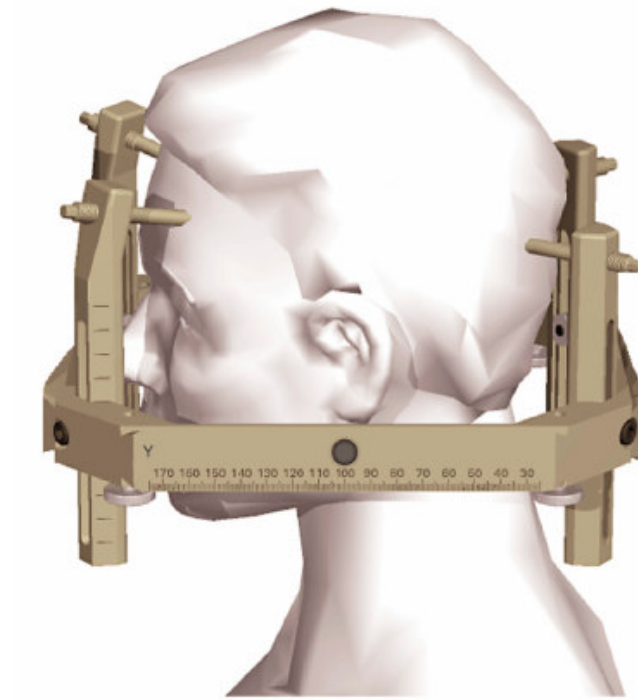


Figure 8 Leksell Stereotactic Frame



Figure 9 Leksell MRI Fiducial Coordinate Box

configuration. During imaging the aluminum wire or cupric sulfate solution superimposes six bright dots on each axial images slice at specific locations. Figure 10 displays an example MR axial slice with the fiducial marks superimposed.

Leksell Gamma Plan registers each image slice from the image study using the fiducials. Once the images are registered, the physicians will identify the disease to be treated on each slice, which in turn localizes the disease in the Leksell coordinate system.

2.1.7 Radiation Dose Distribution Calculation

Once the disease is localized, a treatment plan is developed in Gamma Plan. This is accomplished by placing one or more radiation shots throughout the treatment volume.

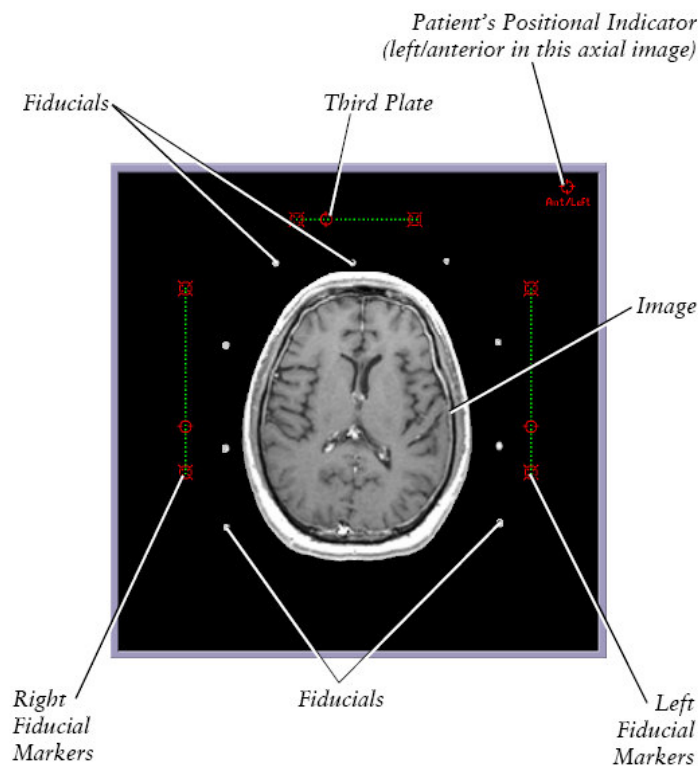


Figure 10 Fiducial System Imposed on MRI Slice

In Gamma Plan the radiation isodose distribution from each radiation isocenter is calculated in a 31x31x31 point dose matrix that surrounds the area to be treated. The spacing between the dose points is varied to cover the treatment volume. Because the isodose lines that are displayed on the image are interpolated from the dose grid, it is ideal to have the smallest spacing possible between the dose points that still encompass the treatment volume.

The radiation transport in Leksell Gamma Plan is performed by calculating the dose contribution from each individual source. Gamma Plan treats the dose delivered from each separate Cobalt-60 source as identical. This allows the dose calculation to be based on measurements of an individual source from each of the four available collimator helmets. The radial dose distribution measurements were performed with a source to focal point distance of 40 cm (the same as the Gamma Knife) with the beam attenuated by an 8 cm water phantom. The relative dose D_i from a Cobalt-60 source is determined

from the following equation

$$D_i = \frac{D_{fi}}{\sum D_{fi}} \left[\frac{d_{f-s}}{d_{f-s} - dz} \right]^2 e^{-\mu dz} P(d)$$

where dz is the distance along the beam axis from the point of focus to the intersection with the perpendicular line to the point, D_{fi} is the relative contribution from source i to the total dose at the point of focus (the term relative is defined as the dose value being equal to 1 at a depth of 1 cm in a water phantom and with the point of focus at a depth of 8 cm), d_{f-s} is the distance from the focus to the source (40 cm), μ is the linear attenuation coefficient of numerical value 0.0063 mm⁻¹ based on tissue equivalent material and a photon energy of 1.25 MeV photon (average of 1.17 and 1.33 MeV) of a Cobalt-60 disintegration, $P(d)$ is the transverse dose at a distance d from the focus. This formula can be broken into

four essential components. $(d_{f-s}/(d_{f-s} - d_z))^2$ is an inverse square factor that accounts for the location of the point of calculation relative to the isocenter of the machine. Absolute dose calibration of the machine is performed at isocenter allowing for a simple inverse square ratio to account for the change in output due to location of the point of calculation relative to isocenter. $e^{\mu d_z}$ is a term that accounts for the attenuation of the radiation beam in tissue. $P(d)$ is a term that accounts for the transverse beam profile, or the off-central axis dose deviation of the beam. $D_{fi}/\Sigma D_{fi}$ is a relative dose term of the dose from a single source to all the sources. The relative dose from source i is calculated

by the following equation $Df_i = \left[\frac{D_{f-s} - 70}{D_{f-s}} \right]^2 e^{\mu(d_{e-r} - d_{f-ei})}$ where d_{er} is the distance between the point of entry of the beam at the skull minus the reference point which, by definition, is set to 1 cm and d_{f-ei} is the distance of the focus point minus the point of entry of beam i at the skull. Figure 11 displays the geometry of the calculation.

2.1.8 Leksell Gamma Plan Treatment Planning

Treatment planning in Leksell Gamma Plan is accomplished by placing one or more radiation shots to cover the area to be treated. The planner has the option of using as many radiation isocenters as necessary placed at any location, any combination of the four treatment helmets, and any gamma angle between 65° and 135°. A gamma angle of 90° is neutral, with greater than 90° being a flexion, and less than 90° being an extension. The size, location, and irregularity of the shape of the target volume, and the proximity of the disease to critical intracranial structures generally dictate the complexity of the treatment plan. Spherical shaped tumors may require only one radiation shot to produce an acceptable plan. Tumors that are extremely irregular may have as many as thirty

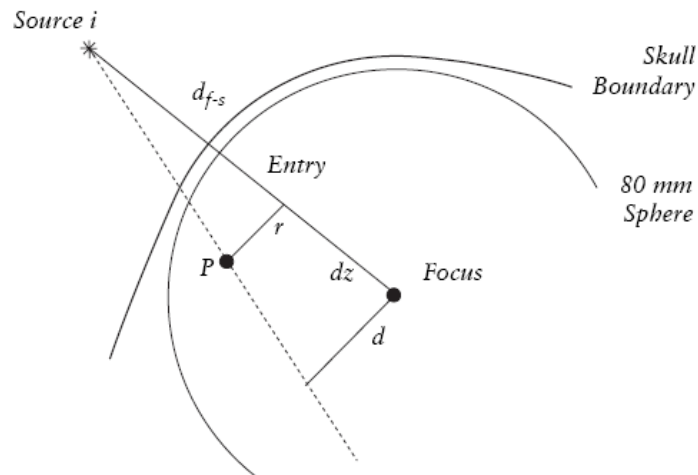


Figure 11 Gamma Plan Calculation Geometry

radiation isocenters with multiple helmets. When multiple shots are used, the interaction of the shots with each other effects the dose distribution. Gamma Plan calculates dose distributions that are relative to the maximum dose point in the dose matrix. The maximum point dose is 100% and all other dose points in the matrix are normalized relative to the maximum point. This is both convenient and problematic. It is convenient because it prevents the planner from creating an area of high dose in the patient greater than intended called a hot spot. It is problematic because any change to the maximum dose point results, in a change to the entire treatment plan. During the treatment planning process, if the weight or location of a radiation shot is changed, and this results in an increase or decrease of dose to the maximum dose point in the calculation matrix, the treatment isodose lines might constrict or extent. This in effect can make a plan inappropriate for treatment. This is best displayed by example.

Figure 12 displays the radiation dose distribution from a single 8 mm isocenter has been placed on a target. The figure displays the distribution on four different MRI slices at four different Z coordinates. The target is the magenta line, the yellow line represents the prescription 50% isodose line, and the green line represents the 95% isodose line, a good representation of the location of the maximum dose point. Note that the 50% prescription isodose line covers the target on the right side of the image well, but does not cover the small area of the target protruding to the left of the image. This lack of dose coverage would be clinically unacceptable, thus a second 8mm isocenter is added to the under-dosed area as seen in figure 13. When the second isocenter is added the maximum dose point is shifted to the left side of the image and the isodose lines are renormalized. The result is adequate dose coverage of the previously under-dosed area, but inadequate dose coverage of the right side of the target in the image and a clinically unacceptable plan. Further refinement of the isocenter position and weights and perhaps the helmets selected is now necessary to adequately cover the target. This renormalization is what makes Gamma Knife treatment planning difficult. Any change to the plan that affects the maximum dose point, will affect the isodose line throughout.

2.1.9 Gamma Knife Treatment Delivery

During treatment the patient is attached to the Gamma Knife via an automatic position system call APS or a manual positioning system known as trunnions. The positioning system coordinates are set to the planned position, the treatment staff leaves the room, and the treatment is initiated from a remote control console. During treatment the shielding jaws open, the couch moves into the Gamma Knife and docks the helmet with the primary collimator. The control console times the treatment and removes the patient

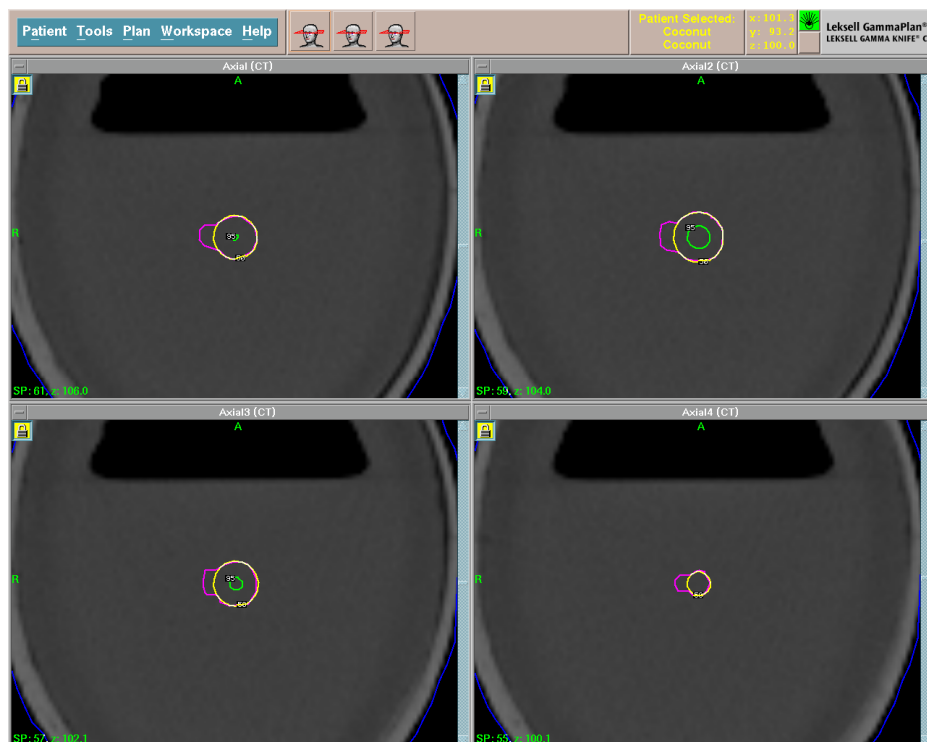


Figure 12 Gamma Knife Treatment Planning 1

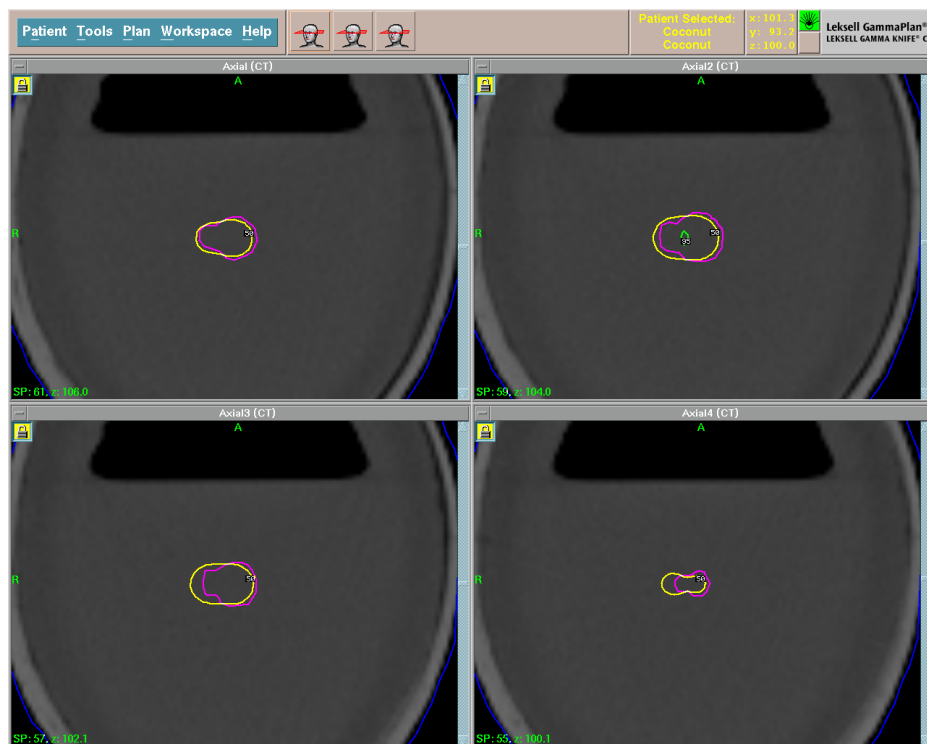


Figure 13 Gamma Knife Treatment Planning 2

from the beam once the planned time is completed. If the treatment has multiple isocenters the APS will move the patient to the next treatment position or when treating with trunnions the staff will manually set the next radiation isocenter position. Once all planned treatment positions are delivered, the patient is unattached from the Gamma Knife and the Leksell frame is removed from the patient. Patient treatment time can be as little as 10 minutes to several hours depending on the complexity of the treatment plan.

2.2 Existing Optimization of Gamma Knife Treatment Plans

Several authors have attempted to automate the Gamma Knife treatment planning process using different approaches. These techniques have varied widely in complexity and the degree to which the process is automated. The majority of developed approaches require some user initialization that usually requires some a priori information concerning the problem. Once initialized, several of these techniques use an optimization method to minimize an objective function. Gradient searches, simulated annealing, multiplier penalty methods, mixed integer programming, non-linear programming, and direct search techniques have all been used to minimize the objective function. Other approaches focus on the actual shape of the target, placing treatment shots throughout the volume based on geometry by a technique termed “skeletonization.” Some techniques feature initialization modules that may determine the initial plan through some data mining module.

2.2.1 Conformity Oriented Approaches

Several authors have formulated the optimization problem with an objective function based on compliance of specified doses to target and normal tissue regions of interest [16,

17, 18, 19]. Yan, et al. formulated their objective function as a weighted combination of a least square error for the target and normal tissue region. Shu, et al. used a multiplier penalty method such that the calculated dose should comply with a set of threshold conditions. Both of these works that are based on the conformity of the dose to the target and non-target regions are reviewed in the following sections.

Clinical Treatment Planning Optimization by Powell's Method for Gamma Unit Treatment System

Yan, et al. [16] developed a system to optimize Gamma Knife treating planning that refines a “roughly” developed plan with Powell's method. Powell's method (also called conjugate direction method) is a non-linear optimization method that does not require a cost function derivative. The method does a quick search for the optimal result depending on the value of the cost function at each iteration. In principal, it is a kind of conjugate direction search method. The method utilizes the sequential search directions in a one-dimension search and generate a new direction towards the next iterative point. The conjugate direction is determined by summing the defined unidirectional vectors. The objective function for the optimization is $f = \alpha \cdot \Delta D_e + \beta \cdot \Delta D_o$, where α and β are the weighting factors for target and normal tissue dose criteria importance, and ΔD_e and ΔD_o are the sum of square errors for the target and normal tissue dose criteria. The radiation dose is computed only at the edge of the treatment target with a primary model in the interest of decreasing optimization time. The dose at a point from one isocenter is

$$D(s) = \sum_{i=1}^{201} M_i \cdot A_i \cdot OF_i \cdot TMR \cdot OAR \cdot F^2$$

where M is a constant to convert to cGy, OF is the

output factor for the specified collimator, TMR is the tissue maximum ratio measured in water (depth dose), and F^2 is the inverse square factor. The dose from multiple isocenters is determined by summing the dose from individual isocenters. The system optimizes the x, y, and z location of the treatment isocenters, the weight of each isocenter, and the size of the treatment collimator for each shot. The number of isocenters is also optimized by evaluating the performance of the optimization at each iteration and adding an isocenter if the result is not satisfactory. If too many shots are initially included, the plan is optimized and shots with low weights are removed.

The authors found the system to converge quickly and not be sensitive to the parameters selected in the initially developed “rough” plan. Additionally, the system was found to achieve an excellent conformation of the estimated isodose curve with the target contours. Potential problems with the system include the computation of the dose at only the edge of the target volumes. Large dose gradients are inherent in Gamma Knife radiosurgery and less than complete dose estimation may lead to less than optimal treatment plans. Additionally, the quality of the “rough” plan will mostly need to increase as the irregularity or complexity of the target volume increases. This method may be best served as refining an already developed treatment plan. Also while Powell’s method is an efficient and reliable zero-order optimization method, situations exist where it may not perform well. For instance, when a search direction cannot provide further improvement, the subsequent search direction will not be conjugate. Also, sometimes the search directions may become parallel after a few iterations [60].

Three-dimensional optimization of treatment planning for gamma unit treatment system

Shu, et al. [17] developed a system for optimizing Gamma Knife treatment featuring a multiplier penalty method to mathematically determine the optimal Gamma Knife treatment parameters. The optimization technique was used by applying a set of threshold conditions; 1) At the target volume points, particularly at definite points on its perimeter, the total dose must be greater than or equal to a threshold L_1 , 2) At any critical structures, the total dose in each of them is less than or equal to the threshold L_2 . To evaluate these criteria the author denotes S_1 the set of the point P_1 selected on the contour of the target volume, S_2 the set of points P_2 in the upper-limit threshold of the critical structure, and S_3 the points P_3 used for the cost function, with the parameters to be optimized denoted as $X=(x_1, y_1, z_1, s_1, w_1, \dots, x_n, y_n, z_n, s_n, w_n)$. Thus the cost function for the optimization is $\min F(X) = \sum_{P_3 \in S_3} I(P_3)$ where s.t. $I(P_1) \geq L_1$, for $P_1 \in S_1$ and

$I(P_2) \leq L_2^j$, for $P_2 \in S_2^j, j=1, \dots, N_s$ where N_s is the number of critical structures

considered in the optimization. Realizing that the total number of points N_p in the set S_1

and $S_2 = 1, \dots, N_s^j$ for $i=1, 2, \dots, N_p$ $g_i(x)$ is defined as $g_i(X) = \begin{cases} L_1 - I(P) & \text{if } P \in S_1, \\ I(P) - L_2^j & \text{if } P \in S_2^j, j=1, \dots, N_s \end{cases}$ and the

cost function is rewritten as $\min F(X) = \sum_{P_3 \in S_3} I(P_3)$ where s.t. $g_i(X) \leq 0$, for $i=1, 2, \dots, N_p$.

A multiplier penalty method is utilized to optimize the cost function. Multiplier penalty methods replace a constraint optimization problem by a series of problems without constraints. The solution of these series of unconstrained problems converges to the solution of the original constrained problem. In minimizing the cost function with

constraints, the corresponding minimization problems are formed by adding a penalty term to the objective function. The penalty term grows when the constraints are violated and is zero in the region where constraints are not violated. The penalty term is usually a product of a positive penalty coefficient and a penalty function. The method is implemented by denoting X as the vector of parameters to be optimized exclusive of the number of isocenters which is optimized separately. The author states that the initial starting point is determined by using a geometric method based on the three-dimensional target shape and volume as well as the spherical dimension representing the shot. They do not elaborate, but other works by the same authors use a rough plan designed by an expert planner utilizing the three-dimensional shape of the volume [16, 19, 23]. With the initial point X^0 set a positive ascending sequence $(c_m)(m=0,1,2,\dots)$ and an initial multiplier $\lambda^0 = (\lambda_1^0, \lambda_2^0, \dots, \lambda_{N_p}^0) \in R^{N_p}$ where R^{N_p} denotes the Euclidean space of dimension N_p . Next the augmented Lagrange penalty function is constructed where

$$L_{c_k}(X, \lambda^k) = F(X) + \frac{1}{2c_k} \times \left\{ \sum_{j=1}^{N_p} [\max(c_k g_j(X) + \lambda_j^k, 0)]^2 - \sum_{j=1}^{N_p} (\lambda_j^k)^2 \right\}. \quad (1)$$

The problem is then optimized without constraint using X^k as a starting point for the optimization problem $\min_X L_{c_k}(X, \lambda^k)$. This problem is solved using a modified Powell's method to optimize both the continuous and discrete parameters. In the program a polynomial interpolation method is used to deal with the continuous parameters such as weight and isocenter location. The isocenter location is confined to within the treatment volume and only positive values for weight are allowed. An enumerative method is used to optimize the collimator helmet size where the cost function is evaluated for all four different values and the one giving the minimum value is chosen. The number of isocenters is optimized separately from the

other parameters. Once the optimization is stopped, the compliance with the objective function is evaluated. If the critical structure objective is not met, an isocenter is removed and the optimization restarted. If the target volume coverage is inadequate, an isocenter is added and the optimization restarted. When the optimization adds another shot, its initial parameters are set according to the number of shots that previously exists in the treatment plan. If only one isocenter existed, the isocenter position is set to the center of the target; otherwise isocenter position is set to the average value of all other existing isocenters. At the end of the optimization, shots with low weights are removed.

In the work the authors presented two cases where the system was employed; one a true clinical case, and second an idealized case. In both cases the system produced plans that were superior to manually produced plans in treatment volume coverage and critical structure sparing. The optimization routine took 45 minutes to optimize the first case feature four isocenters and 1 hour to optimize the second case feature five isocenters. The authors concluded that the system can serve as a good clinical tool for inexperienced planners, and should assist experienced planners. They attribute the long optimization time to the complexity of the dose calculation and feel a faster computation is necessary.

Conformity Oriented Approached Summary

Conformity based methods featuring conjugate gradient and penalty multiplier methods have several advantages for optimizing Gamma Knife treatment planning. The objective functions used in both methods are fairly simple and should effectively represent a “conformal” Gamma Knife treatment plan. Also, the optimization methods are flexible, simple, and are widely used in optimization. The major drawbacks are long computation

time due to the large search area. In addition, conjugate gradient techniques are susceptible to local minima and may not locate the global minima for the search space. Another major problem with these techniques is the user is required to know the number of shots necessary for the treatment and develop a good initial treatment plan prior to optimization. Often it is difficult to have this a priori knowledge concerning the plan without some form of an initialization module.

2.2.2 Simulated Annealing

Simulated annealing is a commonly employed search method for inverse radiation therapy planning [20]. It is a robust method capable of locating a solution close to the global minimum of a cost function in a large search space. The method is susceptible to slow convergence. Several authors used simulated annealing [21, 22] to optimize the Gamma Knife treatment planning and in some cases in combination with gradient descent techniques [23, 24]. The objective functions for these works have included weighted conformity sum of square errors functions and probabilistic functions that are linked to each voxel in the treatment volume. Two of these works are reviewed in the following sections.

Treatment planning optimization by quasi-Newton and simulated annealing methods for gamma unit treatment system

Shu, et al. [23] developed a two part system for optimizing Gamma Knife treatment planning featuring a finite difference quasi-Newton method and a simulated annealing method. The authors used a two part system to enable optimization of both continuous and discrete parameters. With the treatment volume defined by the physician, the

approximate number of treatment isocenters, collimator helmet, and location of the isocenters is determined by the treatment team. The procedure is done manually based on the three dimensional shape of the treatment volume. This approximate plan is used as the starting point of the optimization. The dose calculation for the system is determined throughout the entire treatment area. The model utilizes output factors, tissue maximum ratios, off axis factors, and the inverse square factor. The cost function for the optimization is a sum of square error between the calculated dose and the desired dose

$$f = \alpha \sum_{i=1}^{N_T} \Delta D_T(P_i) + \beta \sum_{i=1}^{N_N} \Delta D_N(P_i), \text{ with } \alpha \text{ and } \beta \text{ as weighting factors for the degree of}$$

desirability dose to target and normal tissue points, and ΔD_T and ΔD_N the difference between the calculated and desired dose for the target and normal tissues. The $\alpha \beta$ ratio is set according to the proximity of the target volume to the normal tissues. The first part of the system seeks the minimum of the cost function by optimizing the continuous parameters including isocenter position and shot weight. The result of the first part serves as the initial configuration for the simulated annealing method. The second part optimizes the discrete variables such as the number of shots and collimator helmets as well as the continuous variables. The quasi-Newton is a steepest descent method that converges quickly to the exact solution and requires calculation of the first derivatives of the objective function. The first derivative is approximated by a forward difference formula. The initial starting point is set with the developed approximate plan. The discrete variables are made invariable for this part of the optimization. The derivative,

$$\nabla f(X^k) = \left(\frac{\partial f}{\partial x_1}(X^k), \frac{\partial f}{\partial y_1}(X^k), \frac{\partial f}{\partial z_1}(X^k), \frac{\partial f}{\partial w_1}(X^k), \dots \right), \text{ is calculated using the forward difference formula}$$

$$\frac{\partial f}{\partial v_1}(X^k) = \frac{f(X^k + h_i e_i) - f(X^k)}{h_i} \text{ where } e_i \text{ is the } i^{\text{th}} \text{ unit vector of the Euclidean space. Finally}$$

the search direction is constructed $p_k = -H_k \nabla f(X^k)$ and a line search method is used to optimize the problem. The process is repeated until a stopping condition is met.

Once the quasi-Newton optimization is stopped, the optimized plan is set as the initial parameters for the simulated annealing optimization method that is used to improve on the initial optimization. Simulated annealing optimization is necessary because the quasi-Newton method can only be applied to the optimization of the continuous variables in this work. Also, the result of the quasi-Newton may not be satisfactory in certain cases, particularly when the target volume is large and the shape is irregular and many treatment isocenters are required.

Simulated annealing is a global optimization method that simulates the physical process of annealing where a single trial solution is modified at random. The quality of the solution is represented by a variable called energy of the system. The method determines the optimal solution by minimizing the energy of the system. The optimization parameters are varied at random and changes that lead to a lower energy are always accepted and an increased energy is probabilistically accepted. The probability is given

by $e^{-\frac{\Delta E}{T}}$ where ΔE is the change in energy and T is a parameter called the temperature.

Initially the temperature is high corresponding to a liquid or molten state where large changes are possible and it is progressively reduced using a cooling schedule thus allowing smaller changes until the system solidifies at a low energy solution. The simulated annealing optimizes all treatment parameters including isocenter position, weight, helmet size, and number of shots. The same cost function used in the quasi-Newton optimization is used in the simulated annealing optimization. The cost of the

initial parameters configuration is evaluated, the parameters are then randomly altered, the new cost is evaluated, and compared to the initial cost to determine ΔE . The Metropolis test is then applied, i.e. if the cost is reduced, the new parameter configuration is retained, otherwise it is retained with probability $e^{-\frac{\Delta E}{T}}$. The initial value of T_0 is chosen so that approximately 10% of positive ΔE are accepted. T is then decreased with iteration n as $\frac{1}{\ln(n)}$. Generally about 300 iterations are evaluated. Due to the allowable parameter values the position, collimator size, and isocenter weight are varied randomly with positive and negative grains. Each parameter has assigned grain sizes with the grain for the collimator size set to 4.0, the grain for the weight set to 0.2, and the grain for the weight set to 0.02. Collimator size is between 4.0 and 18.0 so only grains that keep these values within the limits are allowed. Also, isocenter weight can not be negative so if the weight is zero only positive grains are allowed. Finally, the modification of the isocenter position does not permit an isocenter center outside of the treatment volume. When the optimization adds another shot, its initial parameters are set according to the number of shots that previously exists in the treatment plan. If only one isocenter existed, the isocenter position is set to the center of the target; otherwise isocenter position is set to the average value of all other existing isocenters. The collimator size is randomly chosen, and the weight is set to one. The simulated annealing optimization is stopped if the final temperature is reached, or the cost value has not changed for three consecutive different temperatures.

The system was found to give a satisfactory result and can assist the Gamma Knife treatment team in designing a clinically acceptable radiation treatment plan. The initial

quasi-Newton optimization was found to rapidly achieve a clinically acceptable plan if the initially selected number of shots and the initial parameters were well selected. It was noted that it can be difficult to initially select the number of shots and isocenter position, especially if the treatment volume is large or irregular. In most cases it is not possible to select a good starting plan without an expert planner spending a significant amount of time; however the author felt this optimization step was useful due to the rapid convergence speed and it can give a good starting point for the simulated annealing optimization. The simulated annealing method was then applied to improve on the optimized plan. The authors found this optimization particularly useful as it was able to adjust the discrete parameters such as collimator size greatly improving the quality of the treatment plan. The total optimization time for the system was reported as 12 minutes (4 minutes quasi-Newton, 8 minutes for simulated annealing) for a simple two isocenter treatment plan and 1 hour and 35 minutes (35 minutes quasi-Newton, 1 hour for simulated annealing) for an eight isocenter plan. Specifications were not listed for the optimization computer.

Optimization of Gamma Knife treatment planning via guided evolutionary simulated annealing

Zhang, et al. [22] developed a semi-automatic system for generating optimized Gamma Knife treatment plans. The system simulates an optimal treatment planning criteria with a probability function that is linked to every voxel in the treatment volume region of interest. A manually developed initial Gamma Knife plan is generated that is then optimized utilizing a guided evolutionary simulated annealing (GESA) algorithm.

After the Gamma Knife treatment team defines the treatment volume and critical structure regions of interest, the team specifies a series of parameters that converts the treatment criteria to mathematical constraints that simulates the treatment criteria. The probability of a therapeutic benefit is defined by $P_{ijk}^+ = \frac{e^{K_{ijk}\Delta d_{ijk}}}{e^{K_{ijk}\Delta d_{ijk}} + e^{-K_{ijk}\Delta d_{ijk}}}$ where $\Delta d_{ijk} = (D_{ijk} - D_{PD}) / D_{PD}$, D_{ijk} is the dose calculated from the shot packing plan for the ijk^{th} voxel and D_{PD} is the prescribed dose, and k_{ijk} is a scaling factor that is positive for tumor and negative for critical structures and imposes the importance of fitting the desired dose to the voxel. The therapeutic benefit probability is calculated for each voxel and is used in the objective function to be minimized such as $E = -\frac{1}{N} \sum_{ijk} (P_{ijk}^+)^2$ where N is the number of voxels in each region of interest.

Given the shape and volume of the target the treatment team determines the maximum number of shots and a collimator helmet size for each shot. The weight and position of each shot is then randomly set within the parameters domain. The manually generated plan is then optimized via GESA. The GESA begins by randomly choosing N parents from the search space. Each parent generates M children with a small random difference between the parent and the children. The smallest random displacement of the shot from parent to child is 0.1 mm and the smallest change in weight is 0.01. Also, in the reproduction step the collimator helmet size can also mutate if the weight of the shot falls under 0.3. When this happens the size of the shot changes to the next smaller size and the weight of the resulting shot is assigned as $w_{mutated} = w_{current} \frac{size_{current}}{size_{mutated}}$. A 4 mm shot

whose weight falls below 0.3 is removed from the plan. Once the children plans are created the objective function is evaluated for each new plan. The child plan is accepted into the next generation if it satisfies $e^{-|x_c - x_p|/T_3} > p$ where x_c is the child x_p is the parent, T_3 is a temperature parameter and p is a random number between 0 and 1. The optimization features two levels of competition; children vs. parent and family vs. family. The best child competes against the parent. The objective value of each child is computed, and the best child compete against their parents for inclusion in future generations by $e^{(y_{bc} - y_p)/T_1} > p$ where y_{bc} is the objective value of the best child, y_p the objective value of the parent, T_1 a temperature parameter, and p a random number between 0 and 1. Families compete against each other by determining the family with the most children evolving nearer the best current solution. Families that have more children surviving are likely to include yet more children in succeeding generations. The three temperature parameters decrease exponentially between generations and are assigned initial values of 5.

The GESA system was used to develop a treatment plan for two acoustic tumors. In both cases the GESA was able to produce a conformal plan with similar normal tissue sparing and tumor coverage. The optimization ran for 40 generations and took approximately 1.5 hours to complete compared to 1 hour for the manual planning team. In both case the GESA system reduced the number of shots compared to the manually developed plan. The system needed an initial input of shot number and size. They noted a good initial condition might be produced from a medial axis guided method (reviewed in section 2.2.5) and may improve optimization time. Future work included adapting the code for

parallel processing to improve speed and incorporating an option to generate a dose-shaping plugging pattern to improve conformality.

2.2.3 Mixed Integer Programming

Integer and combinatorial optimization deals with minimization of problem with many variables subject to inequality and equality constraints [25]. A mixed-integer program is a type of optimization of a linear function subject to linear constraints where some or all of the variables are required to be integers. Mixed Integer programming has been use for Gamma Knife radiosurgery [26] and LINAC based radiosurgery optimization [27]. The following section reviews the application of mixed integer programming to the Gamma Knife treatment planning problem.

Optimization of Gamma Knife radio- surgery, in Discrete Mathematical Problems with Medical Applications

In this work, Ferris et al developed a Gamma Knife planning optimization system that modeled the radiation dose calculation and then optimized the treatment planning via two methods; mixed integer programming and nonlinear programming. The non-linear methods will be reviewed in section 2.2.4. The radiation dose distribution is determined by a sum of the contribution of each shot in the plan through an algebraic model of the helmet. This is the same dose formalization reviewed in section 2.2.4. Two objective functions were utilized in the system; the first imposes a constraint on the minimum isodose line that must surround the target volume, the second is a conformity constraint. The conformity constraint specifies that at least a specified percent of the total dose must be deposited in the target and places at upper bound on the dose to the target. Thus the

optimizer seeks to minimize the total under-dosage in the target. A target pixel would be considered under-dosed if it receives less than the prescribed isodose. Both of these formulations assume the treatment team is able to determine the total number of shots necessary for the plan a-priori based on the size, shape, and irregularity of the target volume.

As optimization techniques for solving nonlinear mixed integer programming problems are limited, the authors restricted their work to linear models and thus needed to ensure that the dose calculation constraint is linear. This is accomplished by determining all possible combinations of shot locations and calculating the dose at each pixel location in the target. The optimization algorithm then decides whether or not to use a shot at a particular location. The resulting problem mixed integer problem is then expressed as

$$\min \left(\sum_{(i,j,k) \in R} \text{Dose}(i,j,k) \right) \text{ subject to } \text{Dose}(i,j,k) = \sum_{(s,w) \in \mathcal{G}_S \times \mathcal{W}} t_{s,w} D_{s,w}(i,j,k),$$

$$1 \leq \text{Dose}(i,j,k) \leq 2, \forall (i,j,k) \in \mathcal{T}, 0 \leq t_{s,w} \leq \bar{T} \psi_{s,w}, \psi_{s,w} \in \{0,1\}, \text{ and}$$

$\sum_{(s,w) \in \mathcal{G}_S \times \mathcal{W}} \psi_{s,w} \leq n$. The optimization model was written in the GAMS modeling language with the CPLEX optimization package used for the mixed integer programming optimization.

A major drawback to calculating the dose at all possible locations is the enormous amount of data that is required. The amount of data necessary to perform the optimization is reduced by only calculating the dose at the rind (volume immediately inside and outside the target) of the target volume, rather than throughout the entire

volume. The data load is also reduced by reducing the number of possible shot location. This reduces the number of integer variables $\psi_{s,w}$ (typically found to be the limiting computational factor), and reduces the amount of pre-calculated data required for the optimization. The number of shot location is reduced by creating a coarse grid of large shots, and then randomly placing smaller shots near the boundary of the target until a pre-specified number of shots is generated. Also, the system is inherently less robust than other optimization routines reviewed as the optimizer is not allowed to choose the shot location, but only rather if the shot is included in the plan.

The success of the mixed integer optimization approach and the amount of time to run the problem is largely dependent on how restricted the number of possible shot locations is made. By restricting the grid to much, the resulting constrained problem is infeasible. Further more restriction requires a fast MIP solver and time. However the major benefit is the approach is guaranteed to find a global optimal solution for the particular choice of shot possibilities and grid size.

2.2.4 Non-linear programming

In mathematics, nonlinear programming (NLP) is the process of solving a system of equalities and inequalities, collectively termed constraints, over a set of unknown real variables, along with an objective function to be maximized or minimized, where some of the constraints or the objective functions are nonlinear. [20, 28] Several authors have applied non-linear programming to the Gamma Knife planning problem [29-36]. The following sections review two of those works.

Inverse treatment planning for Gamma Knife radiosurgery

Shepard, et al. [28] developed a Gamma Knife optimization system where the radiation dose distribution is calculated from a model of the helmet dose profiles and a formal constrained optimization is used to produce the radiation treatment plan. The technique optimized the shot size, location, and weight. The optimization does not require a starting plan developed prior to the optimization. The radiation dose is determined from

an algebraic model of the helmet $\sum_{i=1}^2 \lambda_i \left(1 - \operatorname{erf} \left(\frac{d-r_i}{\sigma_i} \right) \right)$ where $\operatorname{erf}(x^i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x^i} e^{-x^2} dx$ and

d is the distance from the center of the shot, and the parameters λ_i , r_i , and σ_i are determined for each helmet via a least square fit of the radiation profile. This algebraic model should reduce calculation times and allow the optimization to be performed more quickly, an important requirements for any Gamma Knife optimization system. The author noted that the algebraic fit was best for the smallest helmet and was worse for each subsequent helmet. The constrained optimization models were written in the modeling language of GAMS (the General Algebraic Modeling System) and the constrained nonlinear programming problem was solved using the CONOPT algorithm. CONOPT uses first-order derivative information to determine a search direction, which is then investigated using a classical line search technique. A feature of CONOPT is that it maintains a feasible solution throughout the optimization once a feasible solution has been obtained. A feasible solution is defined as a solution meeting all optimization constraints or conditions that must be met. The constraints are used to specify 1) the maximum number of shots, 2) a minimum isodose line that must surround the entire treatment volume, and 3) a maximum dose for each sensitive structure. Once the constraints are met, the algorithm begins optimizing the objective functions. The authors

tested several objective functions including 1) maximizing the conformity of the dose distribution, 2) minimizing the number of shots, and 3) minimizing the dose to sensitive structures. The constraints and objectives are each used in different phases of the optimization meaning the first phases optimizes the plan to met the constraints and the second phases further optimizes the plan to met the objectives. A non-linear programming approach was used to optimize the radiation treatment plan that incorporates migrating shot locations. This means the position of each shot can be changed over the course of the optimization. At each location a weight is assigned for each of the four available helmets. A shot of each helmet sizes is then placed at the location if the optimized weight is non-zero. It appears each helmet is evaluated at each position. The dose computation is then performed, and each point is categorized as target or normal tissue. The constraints and objectives are then evaluated. The minimum dose to the target volume and maximum dose to sensitive structure constraint is easily determined by the minimum dose to a calculated target point and maximum dose to a sensitive tissue point. The minimum number of shots objective is evaluated by assuring that at least the minimum number of shots have non-zero weights.

The authors concluded that the system was able to improve dose conformity while reducing the total number of shots, thus the tool has the potential to improve both the quality and the efficiency of Gamma Knife planning. They presented three two dimensional results displaying the ability of the algorithm to develop a treatment plan for non-clinical volumes. The system optimized a concave structure near a sensitive structure, a triangle, and a structure in the shape of a musical note. A three dimensional

clinical case was also present comparing the algorithm to an expert planner showing the algorithm developing a superior treatment plan. However, the author noted a significant degrading when the optimized treatment plan was inputted into Leksell Gamma Plan for final dose computation. This degradation was contributed to a lack of fit between the dose profiles in Leksell Gamma Plan and the developed algebraic model. They stated they were unable to exact the profiles directly and this contributed to the lack of fit. The amount of time for optimization ranged from 5-45 minutes on a Sparc Ultra-10 330 MHz processor. The speed was highly dependent upon both the number of shots and the number of calculation points. To improve speed a coarser grid was used for an initial optimization, and a finer grid to refine the result. Future work included improving the dose model and improving the starting point for each optimization. The optimization starting points were randomly selected and an improvement in the starting point may reduce optimization times.

Clinical Implementation of an Automated Planning System for Gamma Knife

Radiosurgery

Shepard, et al. [29] refined the worked summarized in section 2.2.4 made to the inverse planning tool since the publication of the original article. Recall the system experienced a significant degradation when the optimized treatment plan was inputted into Leksell Gamma Plan for final dose computation. This degradation was contributed to a lack of fit between the dose profiles in Leksell Gamma Plan and the developed algebraic model. In this work the dose engine was significantly improved to minimize the discrepancy. Additionally the system now features a guide to assist the planner in the selection of the number of shot and collimator helmet size for the optimization. A geometry-based

heuristic is used to quickly obtain a high-quality starting point for the optimizer reducing optimization time. Finally, the process was made more flexible with the ability to prescribe localized dose escalation and include dose constraints to critical structures.

The dose discrepancy was reduced in this work by modeling the dose cloud as an ellipsoidal rather than a sphere. A sum of error function was used to approximate the

distribution as follows: $D_w(x_s, y_s, z_s, i, j, k) = \sum_{p=1}^2 \lambda_p \left\{ 1 - \operatorname{erf} \right.$
 $\times \left[\frac{\sqrt{(i-x_s)^2 + \mu_p^y(j-y_s)^2 + \mu_p^z(k-z_s)^2} - r_p}{o_p} \right] \left. \right\}$. The geometry-based heuristic is designed to quickly produce a reasonable configuration of shot sizes and locations with the resulting plan serving as the starting point for the non-linear programming optimization. In the system each shot is modeled as a sphere and the medical axis transform of the target volume is used to guide the placement of the shots. This system differs from other “skeleton” systems (reviewed in section 2.2.5) in that it uses a morphologic thinning approach to create the skeleton as opposed to the Euclidean distance technique. The shots are placed along the skeleton of the target and the algorithm modifies the configuration of shots to provide the best tumor coverage. The geometry based heuristic also servers to guide the user in selecting the number of shots and the most appropriate collimator helmet, a sort of intelligent initiation module. The user is presented several shot combination options that should “geometrically” cover the treatment volume. The user then selects the shot option based on plan quality vs. treatment efficiency. Once an initial plan is select the geometry based optimization is used as the starting point for the non-linear optimization discussed in section 2.2.4.

The authors compared the treatment plans produced by the optimization system and an experienced planner for ten cases. On average the inverse planning tool provided improved conformity, higher minimum target dose, and reduced volume of the 30% isodose line. The system typically optimized the treatment plan in less than 10 minutes on an 800 MHz PC.

2.2.5 Geometry Optimization Approaches

Several authors have utilized a geometric based optimization approach for radiosurgery treatment planning optimization sometimes termed “skeletonization.” [37-48] The following section reviews one such works.

Morphology-guided radiosurgery treatment planning and optimization for multiple isocenters

Qu, et al. applied principals from 3D morphology in this work to solve the problem of 3D treatment planning and optimization in stereotactic radiosurgery. The work uses the medial axis transformation, or “skeleton”, of the target volume that uniquely characterizes the target to determine the optimal shot positions, helmet sizes, and the total number of shots necessary to cover the target. This approach reduces the complicated 3D optimization problem to a 1D searching problem.

Skeletonization is a description of an object’s shape that extracted both morphological and volumetrical features of the object. The skeleton retains the topology of the original shape and the original shape can be reconstructed from its own skeleton. The skeleton

can be visualized by considering a simple example described in figure 14. Suppose we would like to determine the skeleton of the rectangle in the upper left of the figure. The skeleton of the rectangle can be determined by moving the entire rectangle boundary inward at a constant rate. When two boundaries overlap by some defined cutoff distance the intersection defines a point on the skeleton. When all boundaries completely overlap the skeleton will be completely formed. The skeletonization may result in several “branches” that combine to form the entire skeleton. The intersections of branches are known as crosspoints, and the end of a skeleton branch is known as an endpoint. Each point on the skeleton is the bisector of at least two volume boundary points, i.e. the skeleton points compose a 2D surface if it is the bisector of two or three boundary points

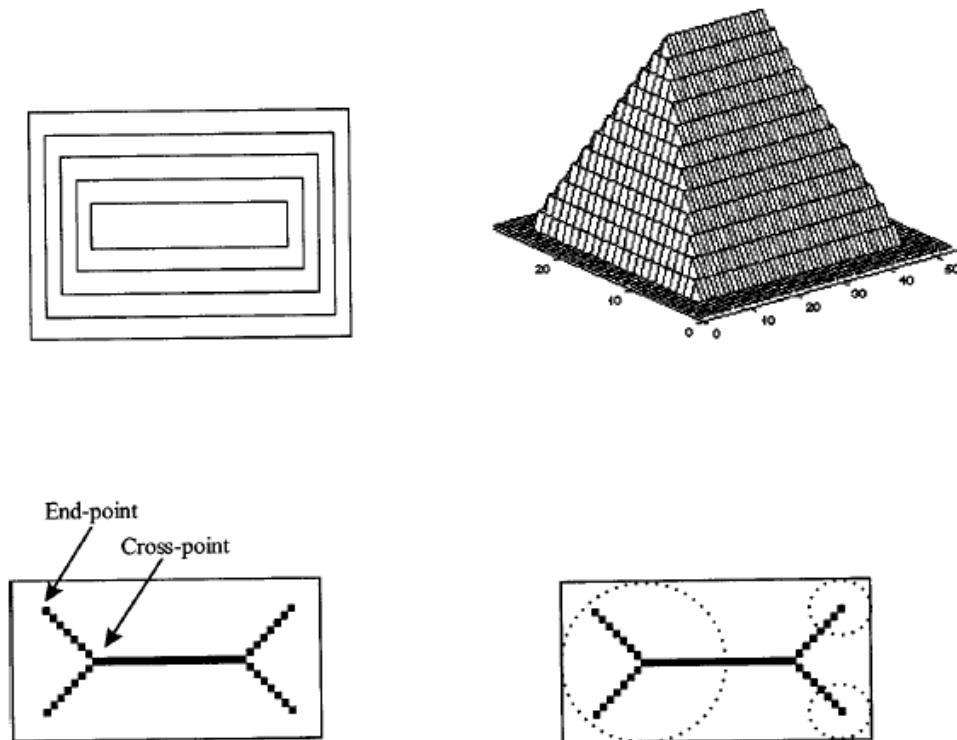


Figure 14 Skeletonization Example

and compose a 1D string if it is the bisector of three boundary points. Each skeleton point is associated with a radius value representing the exact distance from the bisector point to its nearest boundary points. This radius is the maximum internal disk that fits into that region. If a radiation shot is located on the skeleton and the shot size is the associated maximum internal disk, the shot can optimally cover that part of the target volume. In this work, the skeleton points provide optimal shot positions and the radius associated with the skeleton point is the optimal size for the shot that can be calculated as a combination of the collimator size and relative weight.

The mathematical formulization of the 3D skeletonization is determined via a simulated dynamic grassfire computer-generated by a vector distance transformation. A skeleton point is detected by measuring the maximum curvature of the dynamic grassfire map at each pixel/voxel. The curvature K at pixel p is defined as $K = \left| \frac{d\vec{t}}{ds} \right|$ where s is the arc length parameter and $\vec{t}$ is the unit vector tangent to the wave at p . The tangent to the curvature of a 3D surface at a voxel is not uniquely defined, thus there exist an infinite number of grassfire waves and hence an infinite number of tangent vectors to the curve. For each of these tangent directions there is one curvature value and among all tangent directions there are two characteristic directions along which the curvature is minimum and maximum. The characteristic direction at each point and the curvature extremes can be obtained from the Hessian matrix. By denoting V_1 , V_2 , and V_3 as the eigenvectors of the Hessian matrix with corresponding eigenvalues λ_1 , λ_2 , and λ_3 a surface curvature at

point P can be expressed as $K_s = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = - \begin{bmatrix} V_1^T \cdot Hess(f) \cdot V_1 \\ V_2^T \cdot Hess(f) \cdot V_2 \end{bmatrix}$. K_s describes the

curvature characteristics of the wave surface at a point, or the measured deviation of the local wave surface from flatness. V_1 represents the direction in 3D along which the local wave surface has the maximum curvature, and V_2 is the direction along which the local wave surface has the minimum curvature. Next using the surface curvature index concept the authors developed a new algorithm to search the skeleton points in both 2D and 3D and used a final thinning step to reduce the skeleton to a single pixel width.

The optimization criteria are 1) the dose covers the entire target region to within a tolerance ϵ , 2) the optimization should use the least number of shots possible, and 3) all shots should be confined with the target volume without overlap. The treatment plan optimization is formulated mathematically as $\min N | P_N(R), \quad \forall S^{d_i}(x_i) \subset R, \quad i=1 \text{ to } N,$

with $\frac{V(R) - V[\cup_{i=1}^N S^{d_i}(x_i)]}{V(R)} \leq \epsilon$ where P denotes the plan, R is the target volume, and $S^{d_i}(x_i)$ is the shot location with diameter d_i . For optimal coverage of the target R , the center of each shot must be located on the corresponding skeleton $X(R_i)$ where R_i is the residual region after $S^{d_1}(x_1), \dots, S^{d_{i-1}}(x_{i-1})$ has been segmented from the original region.

Denoting $P_{N_{opt}}(R)$ as the optimal coverage of the target region R then there exists an optimal configuration of shot with varied sized and placement of the skeleton $X(R_i)$ that

covers the region R expressed as $P_{N_{opt}}(R) = \bigcup_{i=1}^{N_{opt}} S^{d_i}(x_i) = P_N(R) | N = \min(N), \forall N,$ where

$R_i = R - \bigcup_{j=1}^{i-1} S^{d_j}(x_j)$. If the optimal plan only considers shots centered on the skeleton, the resulting radii are always optimal, hence the position and size of a shot match the region

simultaneously and the complexity of the optimization is thereby reduced to a 1D search problem. The optimal coverage is finally determined via a global combinatorial search process expressed as $\min_N |P_N(R) = \min_N |S^{d_i}(x_i) + P_N(R - S^{d_i}(x_i))|$
 $= 1 + \min_N |P_N(R - S^{d_i}(x_i))|, \forall S^{d_i}(x_i) \subset R$. In the optimization, weights are limited to values of 0.7-1.0, the shot resolution is limited to 0.3 mm, and the helmet size is allowed to vary to all available helmets.

The authors presented an example optimization of a 40 mm width target. The skeletonization of the target resulted in three branches. The authors limited potential shot locations to the endpoint and crosspoints as these points provide maximum coverage to multiple boundaries. This logic should result in the fewest points possible but not necessary the most conformal plan. The optimization tests various combinations of helmet size and weight at each location until an optimal plan was found that conformed to the volume.

The skeletonization method of Gamma Knife optimization has several advantages most notably rapid optimization time. Rapid optimization is a result of the 1) the shots position and size are coupled, 2) only end points and cross points are allowed as shot locations, and 3) dose is not calculated. Also, the method in a sense mimics the method an expert planner would go about producing a plan. However, some of the time saving advantages can result in less than conformal plans. In most instances the optimal shot location would not occur at a cross point of endpoint, but rather several shots surrounding these points. Also, the lack of a dose calculation can results in plans that are optimized

geometrically but not dosimetrically. Because only four different sized collimators are available, in many cases it will be impossible to geometrically cover the volume. The authors state this should be a concern because the actual dose profiles are not step functions, and there is some addition of the lower isodose lines from each shot to the areas that are geometrically under-dosed. However in practice, the penumbra for the Gamma Knife beam is very sharp and inadequate geometric coverage of the target generally leads to inadequate dosimetric coverage. This method of optimization might serve as a good initialization module, however an optimization of the actual dose distribution should be necessary to produce clinically conformal treatment plans.

2.2.6 Data Mining Approaches

Exclusive of the geometry based systems, Gamma Knife treatment planning optimization is largely dependent on a quality starting point. A quality starting point is generally produced by an expert planner who uses their experience of similar cases to guide the production of the initial plan. Currently this knowledge is stored in the planner's brain and cannot only be easily transferred. A database of past plans could be considered a representation of an expert planner's knowledge that could be data-mined to create the initial plan.

Automated gamma knife radiosurgery treatment planning with image registration, data-mining, and Nelder-Mead simplex optimization

The aim of the work by Lee, et al. [49] was to show that treatment parameters from a similar previous treatment plan can be used as a starting point for subsequent computer optimization. When paired with an optimization module this would produce a fully

automated method of performing Gamma Knife treatment plan. The author's collected 1316 treatment plan calculated over 5 years for the Gamma Knife. The collected plans were reflected about the sagittal plan to exploit left-right symmetry of the system giving a total number of 2632 plans in the database. In the Leksell Gamma Plan system, a lesion is contoured manually by the treatment team creating a two-dimensional polygon representation of the region of interest. A three-dimensional mesh representation of the region of interest was then produced using image registration techniques. This mesh was compared to the database to find a similar past treatment plan. The comparison was executed by converting the mesh being matched and each mesh from the database to a binary image. These binary images were then compared by performing an exclusive-OR between the images. The resulting image represents those voxels which belong to at least one but not both images. The total number of such voxels (multiplied by the voxel volume) was converted to a volume then divided by the surface area of the database lesion to give a length. The resulting length, termed the "Barber distance", is a measure of the average distance between the surfaces of the two lesions and considered a measure of the similarity of the two mesh structures. Several treatment plans from the database with the minimal Barber distance were selected and translated to the target lesion by matching the coordinates of the center of gravity of the database selected and current treatment plans. The translated treatment plans were then utilized as the starting point for the optimization.

Optimization is performed using a Nelder-Mead simplex method, a type of direct search that does not require calculation of gradients. In this method a multidimensional simplex,

or polytope, with $n + 1$ points (where n is the number of variables) is formed. Initially, one of the points of the simplex is the best estimate of the n variables, the vector P_0 . The other points are perturbed versions of P_0 , thus for n isocenters the simplex has $4n+1$ points. To derive the points of the initial simplex, variables that were coordinates were perturbed by 5 mm and weights variables were increased by 1.5 times. The cost function is then compared for every point, and a set of heuristic rules applied to move the points in a series of steps, such that at every step the point with the highest cost function is moved to a lower value. The cost function for the optimization is a measure of the

conformity of the dose distribution to the target such that $Cost = -\left(\frac{TV_{PIV}}{PIV}\right)\left(\frac{TV_{PIV}}{TV}\right)^n$

where n is an exponent, TV is the target volume, PIV is the volume covered by the prescription isodose, and TV_{PIV} is the volume of the target covered by the prescription isodose.

The developed system was applied to ten acoustic neuroma and ten meningioma cases and compared to manual developed plans. In most cases the computer optimized plan performed as well if not better than the manual plan, however the system had several shortcomings. The authors stated that the Barber distance, the metric for database comparison, is a crude method that does not strongly correlate in detail with the average normal distance. The metric was chosen for its rapid calculation and comparison to the database; however other metrics such as number of isocenters or pathology may need to be considered in initial plan selection. In addition, the presented method does not permit protection of critical structures in the optimization. Finally the presented work utilizes a full dose computation at each iteration to maintain compliance with the Leksell Gamma

Plan dose computation. This results in unacceptably long computation times. The authors say that future work will investigate the use of dose models to speed up computation times. In spite of the shortcomings the authors believe that data-mining provides an alternative method of generating initial plans for manual or computer optimization.

2.3 Empirical Modeling of Historical Plan Database

The proposed work will use a database of past treatment plans to initialize the optimization routines with a set of plans from past treatment volume that are similar to the current treatment volume. The comparison will be performed with an empirical modeling method. The two major classes of empirical modeling methods used for prediction are parametric and non-parametric [66]. Parametric modeling techniques include linear regression, neural networks and other techniques that map relationships in data by optimizing a set of parameters using a training data set. In this technique the predictors are mapped to the response as $y = f(x) + \varepsilon$ where $f(x)$ is a known smooth function that the modeler must determine the appropriate form. Once the parameter values in the smooth function are determined from a training process, the data is set aside and the model's prediction equation is set. As additional historical plans become available they will not be able to be incorporated into this type of prediction method. In order to improve the selection of plans to initialize the optimization method the empirical model must be able to change its parameters based on newly available historical plans. This requires the use of a non-parametric modeling technique.

Non-parametric techniques use the actual training data to understand future predictions and store the training data in a memory matrix. In nonparametric regression, $f(x)$ is some unknown, smooth function and is unspecified by the modeler. A data driven technique determines the shape of the response curve. To illustrate this technique consider the case of a single predictor and response variable. The predictor observation with the most information about the response should be those closest to the location of the point in question. Therefore, a decreasing function of the distance of the predictors from the point in question is used to determine the weights assigned to each predictors. The predictors closest to the point in question receive more weight than those more remote. This enables the non-parametric technique to bias the prediction to those predictors closest to the point in question. Non-parametric techniques are ideal for the inclusion of newly available historical plans as they can simply be appended to the memory matrix. Non-parametric techniques are also advantageous as there are no training times and no need to optimize the parameter values in the model.

Kernel regression is a non-parametric technique used to map a relationship between a set of predictor and response variables by calculating a weighted average of the historical observations. There are three different types of kernel regression architectures (inferential, hetero-associative, and auto-associative) that are characterized by the number and type of inputs and outputs as illustrated in the figure 15 [67]. This work will use an autoassociative kernel regression model to perform the mapping. The technique employed here is actually selection rather than regression. The set of input variables will be mapped to a set of historical plans that are most similar. The selected plans will then

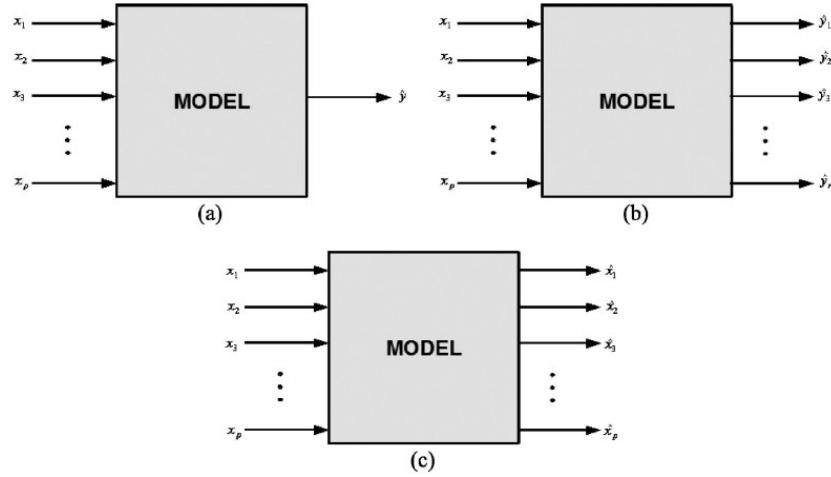


Figure 15 Kernel Regression (a-inferential, b-heteroassociative, c-autoassociative) [66]

be extracted from the database and applied to the current volume as the initial starting point for the optimization modules.

An autoassociative kernel regression model uses inputs to predict the “correct” values for the inputs, where “correct” refers to the relationships and behaviors contained in the exemplar observations. In this work rather than predicting values for the correct inputs, the plans for the exemplars found to be most similar will be extracted. An autoassociative kernel regression is similar to the single predictor example already presented exclusive of the number of input and output variables. The process is illustrated in the following figure 16. A set of input variables are comparable to a historical database containing past cases known as exemplars. The comparison is performed utilizing a simple Euclidian distance measure. For p inputs the distance measure becomes $d(X_b, x_q) = \sqrt{(X_{i,1} - x_{q,1})^2 + \dots + (X_{i,p} - x_{q,p})^2}$. The computed distances are then converted to weights using a Gaussian kernel function. The equation

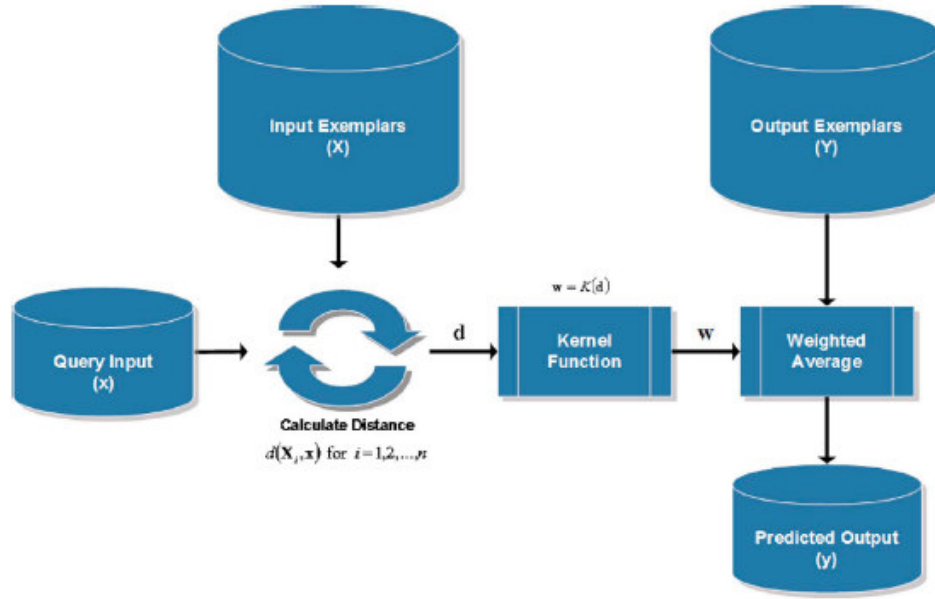


Figure 16 Kernel Regression [66]

for the Gaussian kernel $w = k_h(d) = \frac{1}{\sqrt{2\pi}h} e^{-d^2/(2h^2)}$. The parameter h in the Gaussian

kernel function is the kernel bandwidth and dictates the width of the function. The effect of the size of the kernel bandwidth can be seen in figure 17. As the bandwidth increases the Gaussian kernel function widens, thus the weight assigned to exemplars further from the point in question increases, escalating their affect on the result. The optimal bandwidth is chosen iteratively while comparing each model's prediction error.

In kernel regression after the Gaussian kernel converts the distances to similarities, the estimated similarities are used to perform a weighted average of the output exemplars

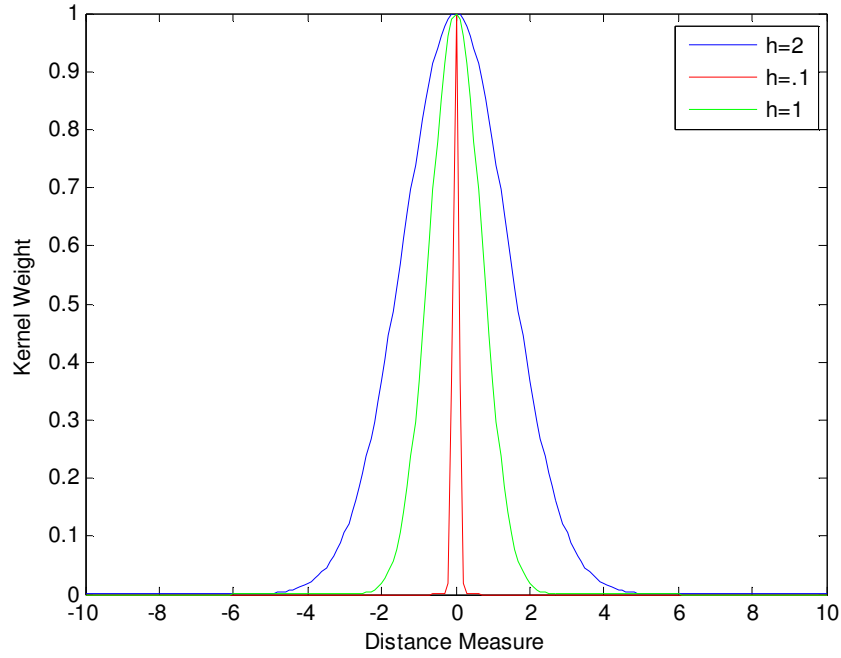


Figure 17 Gaussian Kernel Function with varying bandwidth h [66]

such that $\hat{y}(q) = \frac{\sum_{i=1}^n y_i w_i}{\sum_{i=1}^n w_i} = \frac{\sum_{i=1}^n y_i K(d(x_i, q))}{\sum_{i=1}^n K(d(x_i, q))}$. However, in this work we are not looking

to predict the values of the response, but simply to select the exemplars that are most similar to our current input parameters. Thus the exemplars with the largest calculated weights will be selected for extraction from the historical database and applied to the current treatment volume.

2.4 Genetic Algorithms

Genetic algorithms are an optimization and search technique based on the principles of genetics and natural selection [50-53]. The algorithm is capable of determining the

global minimum of a search space by utilizing a complex system of specified search rules applied to a population comprised of individuals. The individuals are created from a random selection of the parameters in the parameter space. An individual consists of a chromosome and the chromosome calculated cost. The cost is the evaluation of the fitness function of the optimization. The chromosome is an array of parameters that are evaluated and passed on to future generations through mating based on the search rules. Chromosomes can be altered and passed onto future generations during the mating process through crossover (individuals mate and trade chromosomes), asexual reproduction (entire chromosome passed, usually reserved for low cost individuals), and mutation (random mutation of chromosome). Fit individuals are selected for mating, while weak individuals die off. Mated parents create a child with a chromosome set that is some mix of the parent's chromosomes. The genetic algorithm will iterate through some specified number of generations or until it reaches some specified minimum cost searching for the lowest cost of the fitness function. The most successful individual's (lowest cost) chromosomes will be passed to the next generation using the selection rules in a process that emulates survival of the fittest. The process of mating and child creation is continued until an entirely new population of size n is generated with the hope that strong parents will create a fitter generation of children, hopefully leading to an optimal individual or solution. Figure 18 illustrates the genetic algorithm optimization process.

It is important to note that genetic algorithms work with the coding of the parameters, and not the parameters themselves. Binary coding is a type of coding where the parameters are discretized into a number of possibilities. The chromosome length is based on the

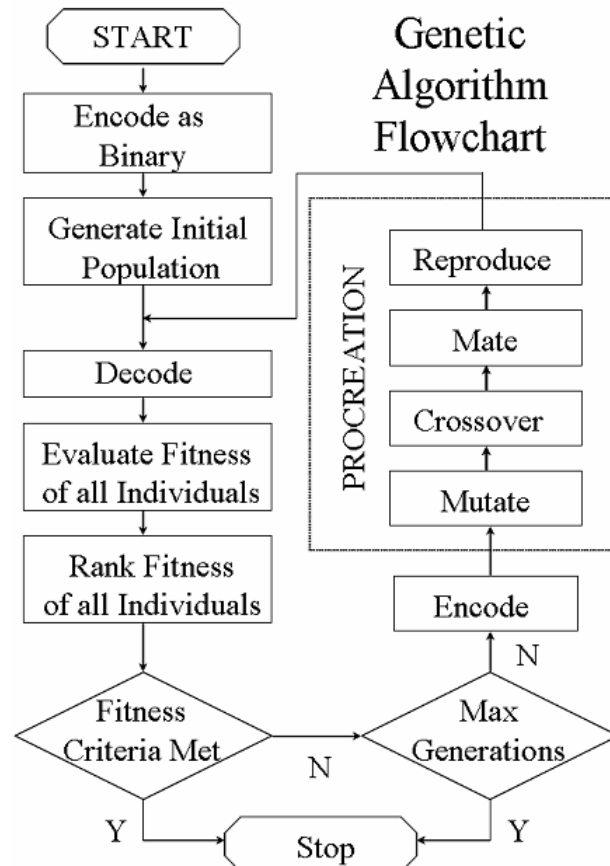


Figure 18 Genetic Algorithm Flowchart

total number of possibilities in a binary format. For example, a string of length 4 would represent 2^4 or 16 possibilities. The result of the mating process between the individuals of the population is determined by a series of mating rules. **Single point crossover** is a process where a crossover point is randomly chosen and the chromosome of the second parent overwrites the chromosome of the first. For example consider a set of parent with chromosomes *abcde* and *ABCDE*. After single point crossover, one possible chromosome for the child is *abcDE*, as the position between the *c* and *D* is randomly determined as the single crossover point. **Uniform crossover** is a crossover method that

each chromosome position has a probability for a crossover with the second parent. It is possible to obtain any combination of the two parent chromosomes. For example the child could end up with chromosome *aBcDe*. **Jump mutation** is a method that produces a chromosome that is randomly picked from a defined parameter range. For example, the child ends up with chromosomes *abcDM*, where *M* was not a chromosome from either parent. **Creep mutation** is a process that produces a parameter value that is randomly picked to be larger or smaller than its parent values. For example, the child ends up with chromosome *abcDF*, where *F* was not a chromosome from either parent, but is only one increment away from second parent's chromosome value of *E*. **Asexual reproduction** is the operator that is used to insure that the chromosome set of the best parent generated to date is reproduced.

Genetic algorithms have been successfully applied to many optimization problems including training neural networks, categorizing large amounts of data, assisting in design, solving the traveling salesman problem, and testing combinatorial solutions such as which group of inputs to an empirical model is optimal [50]. Genetic algorithms advantages include the ability to optimize with continuous or discrete variables, they do not require derivative information, they can simultaneously search from a wide sampling of the cost surface, they can deal with a large number of variables, is suited for parallel processing, and they are capable of handling complex cost surfaces.

A new genetic algorithm technique in optimization of permanent 125 Iodine prostate implants

Genetic algorithms have been used to optimize several radiotherapy problems. Guozhen et al [54-57] used a genetic algorithm for real-time optimization of Iodine 125 prostate brachytherapy. Prostate brachytherapy is a medical procedure where radioactive “seeds” are placed in a grid arrangement encompassing the prostate to treat prostate cancer under trans-rectal ultrasound guidance. Real time optimized treatment planning at the time of the implant is desirable to ensure a conformal, uniform dose distribution in the prostate gland. The dose calculation used in the optimization was the formalization recommended in the American Association of Physicist in Medicine Task Group 43. The cost function used in this study is composed of a) the constraints on minimum peripheral dose at the target contour points, b) the constraints on the dose distribution within the target volume; and c) the constraints on the dose to the critical structure (~e.g. urethra). This formalization is similar to the Gamma Knife optimization proposed in this work exclusive of the dose calculation. In the prostate brachytherapy implementation of genetic algorithm, the coordinates of the possible seed positions were encoded into a fixed-length binary string. At each location on the string, 1 indicates a position for the loading of seeds, 0 indicates a null space. The authors concluded that the genetic algorithm allowed for an efficient and rapid optimization of the dose distribution that was suitable for real time treatment planning. They found that for fixed needle geometry, the optimized plan showed much better dose distribution than that of non-optimized plan. In addition, if the critical structure was considered in the optimization, the dose to the critical structure could be minimized and in the cases of irregular and skewed needle

geometry, the optimized treatment plans were almost as good as ideal needle geometry.

This work demonstrates that genetic algorithms can be successfully applied to a radiotherapy problem similar to the work proposed in this dissertation.

2.5 Hybrid Optimization with Gradient Descent

Genetic algorithms are capable of rapidly locating the region where the global minimum exists, however they take a relatively long time to locate the exact local optimum in the region of convergence [50]. Combining a genetic algorithm with a local search method can speed up the search and at least guarantee locating the global minimum of the region where the genetic algorithm optimization located as optimum. Such a hybrid system can be applied by; 1) running a genetic algorithm until the optimization gain diminishes, then run a local optimizer to further refine the result, 2) seed a genetic algorithm with some local minima found from random starting points in the population, and 3) run a local optimizer every so many iterations on the best or several of the best solutions and adding the resulting chromosomes to the population [50]. A genetic algorithm can be easily coupled to a local optimizer since both use continuous variables. In this work a local search will be applied to the genetic algorithm final solution to converge on optimistically a global minimum.

Gradient descent is a first-order optimization algorithm in which a local minimum is determined by taking steps proportional to the negative of the gradient of the function at the current point. The method requires the gradient of the function can be computed. The size of the step in the negative direction can be either constant or varied. A small step size guarantees convergence but will take longer to run. Some system use a variable

step size that begins with a large step size and shortens the step size later in the optimization to increase efficiency. For gradient descent the first order approximation of the change of the error (ΔE) is: $\Delta E = g^T \Delta x$ where g is the Jacobean gradient vector and x is the step size. A line search can be conducted along the gradient direction to minimize the error E , in turn locating the minimum along this gradient. The process is then repeated along gradient at the located minimum until the minimum of the region is determined as illustrated in figure 19. Note in the figure that the optimization takes a zig zag approach to the minimum.

Gradient descent can be computational intensive and may take a long time to optimize base on the gradient step size, thus this work will use a conjugate gradient method to perform the final treatment plan optimization. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. The method is guaranteed to minimize a quadratic in n steps where n is the dimension of the search. For non-quadratic function as

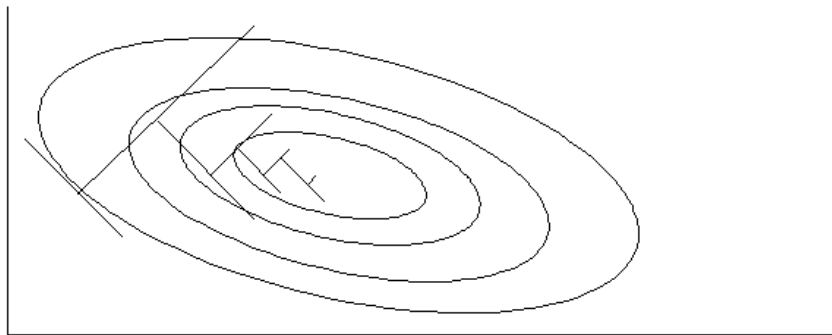


Figure 19 Gradient Descent

is the case in this optimization, the optimization is an iterative process. Movement from the starting point to the minimum point can be expressed as a sum of n steps in directions

p_i of step size α_i expressed as $x - x_{\min} = \sum_{i=1}^n \alpha_i p_i$. The variable p_i are the conjugate

directions that form a basis for the n-dimensional error space represented by $p_i^T H_{p_j}^T = 0$

for $i \neq j$ and $i = 1, 2, \dots, n$ where H is the Hessian matrix of second order partial

derivatives. The first step in the conjugate gradient algorithm is in the direction of the

negative gradient, the same as in gradient descent. All successive steps are in the

direction of the conjugate gradient algorithm where new search directions are a linear

combination of the current gradient direction and the old search direction

$p_{k+1} = g_{k+1} + \beta_k p_k$. Several methods such as Polak-Ribiere and Fletcher Reeves exist to

compute β_k .

3 Methodology

This work uses a genetic algorithm to optimize Gamma Knife treatment planning. An initialization module is used to provide the genetic algorithm with a high quality initial population. The hope is that by providing the genetic algorithm with good starting point treatment plans, the optimization should converge on an optimal result more rapidly. The initial population is determined by comparing the current target volume to a database of historical treatment plans, developed by an expert planner, that have been successfully utilized for Gamma Knife patient treatment. The historical database is composed of target volumes and corresponding treatment plans. The module selects treatment plans from the database by comparing parameters extracted from the current target volume to the database of historical target volumes. The historical target volumes are determined to be similar to the current target volume via a non-parametric modeling technique. The initialization module also produces a set of “rough” treatment plans through a geometrical optimization. This is accomplished by sequentially optimizing the position of individual treatment isocenters until a specified number of isocenters have been added. The position is determined by a genetic algorithm that is attempting to maximize the amount of geometric coverage of a single treatment isocenter. The geometric optimization requires the code user to specify the number of isocenters and they can also specify the maximum collimator size. The user is able to select the number of historical plans to include in the genetic algorithm initial population and the code user has the option to include up to five geometrically optimized plans in the initial population.

Once the example plans are created and applied to the current target volume, the system calls a genetic algorithm optimizer to improve the quality of the plan. The genetic algorithm calls a function that determines the radiation dose distribution for the treatment plans throughout the target volume and critical structure regions of interest. Criteria concerning the quality of each of these treatment plans are calculated from these dose distributions. These criteria include several conformity indices of the radiation dose to the target volume, the percent isodose coverage of the target volume and the maximum dose to critical structures. These criteria are then optimized by the genetic algorithm. The genetic algorithm optimized plan is then further refined by a gradient descent optimizer. Figure 20 is a flow chart outlining the entire automated planning optimization process.

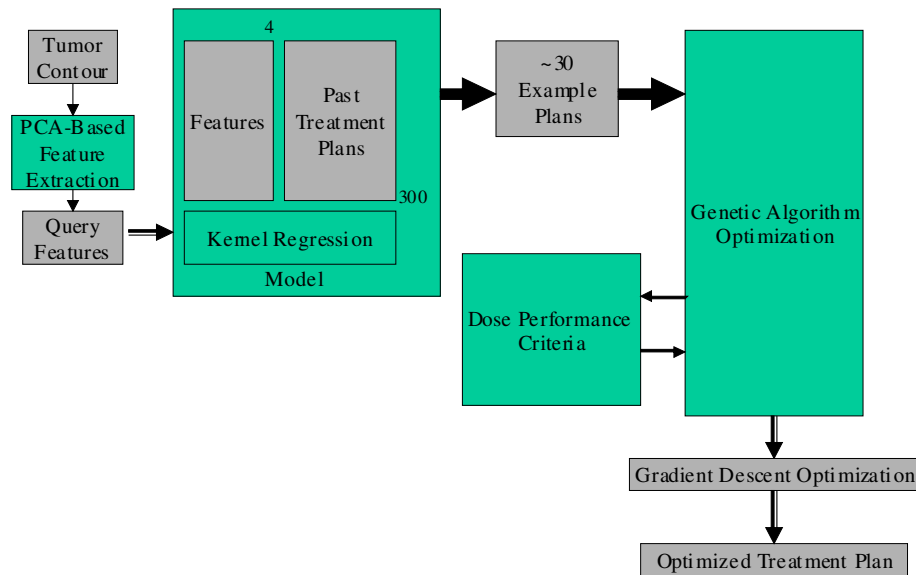


Figure 20 Treatment Plan Optimization Overview

The treatment planning optimization system is coded in the MatLab (**MAT**rix **LAB**oratory) programming language utilizing CERR (Computational **E**nvironment for **R**adiotherapy **R**esearch) as a graphical user interface (**G**UI) for viewing and manual manipulation of the plan. The computer program developed for this dissertation named TOGA (**T**reatment planning **O**ptimization via **G**enetic **A**lgorithms) is utilized for calling the functions developed for this work. MATLAB is a numerical computing environment and fourth generation programming language optimized for matrix computation developed by the MathWorks [58]. All MatLab computer code generated during the course of this work can be viewed in Appedix A: Matlab Code. CERR is a software platform developed at Washington University for sharing research results in radiation therapy treatment planning [59]. CERR is useful in radiation therapy treatment planning research as it a) provides a convenient and powerful software environment to develop and prototype treatment planning concepts, b) serves as a software integration environment to combine treatment planning software written in multiple languages, c) provides the ability to extract treatment plans from disparate planning systems using the widely available AAPM/RTOG archiving mechanism, and d) provides a convenient and powerful tool for sharing and reproducing treatment planning research results. In this work CERR will be used to interact and evaluate the treatment plan. Also, the CERR radiation therapy plan data structure will serve as the data structure for the project. This work was designed to run with CERR 3.3.1 Beta with versions of the code available for CERR 2.6.7 and CERR 3.0.5 Beta.

3.1 Importation of data from Gamma Plan to MatLab and CERR

Gamma Plan is a commercially available software package from Elekta used for Gamma Knife treatment planning. The software package was originally developed to run on a Hewlett Packard UNIX platform. A Red Hat Linux version was introduced in 2007. The software packages are essentially the same regardless of operating system. As outlined in Chapter 2, after the Leksell Model G frame is attached to the patient an imaging study is performed with a Leksell Fiducial Box attached to the frame. This image study may be a MRI, CT, or an Angiogram. The image study is imported into Gamma Plan, the images are registered into the Leksell coordinate system using the fiducial imposed on each image slice, and the patient's anatomy is localized in the Leksell coordinate system. This localization is achieved by the treatment team defining regions of interest on the patient's images in Gamma Plan. The regions of interest may be a tumor, a cranial nerve, the brainstem, a lobe of the brain, etc. Under the current planning process, a treatment plan is manually developed and the quality of the plan is evaluated using radiation dose statistics calculated for the defined regions of interest. The developed importation computer code extracts the treatment images, regions of interest, treatment plan, and calculated radiation dose distribution and converts the data in the CERR data format.

This work is coded to extract the treatment planning data from the UNIX operating system version of the code. Importation is performed by first copying several binary files from the UNIX Gamma Plan computer to the computer running MatLab and TOGA. These files include an image file for each slice of the imaging study, a DoseGroup file that contains the radiation isodose distribution calculated by the Gamma Plan, a DosePlan

file that contains the treatment isocenter position, collimator size, & weight, and a Volume file that contains the regions of interest defined in Gamma Plan for the image study. CERR contains an importation function for reading and converting these Gamma Plan files to the CERR format. The function places the data from these files into a MatLab cell array names planC. PlanC is a multidimensional array with the division of data as follows: (1)header, (2)comment, (3)scan, (4)structures, (5) structureArray, (6)structureArrayMore, (7)beams, (8)beamGeometry, (9)dose, (10)DVH, (11)IVH, (12)planParam, (13)seedGeometry, (14)digitalFilm, (15)RTTreatment, (16)IM, (17)importLog, (18)CERROptions, (19)indexS. The import function did not incorporate treatment isocenter information, thus the function was revised to place this information in the beamGeometry portion of the cell array. This revised CERR function is included in the Appendix A: MatLab Code. This is the only CERR function included in the appendix as it was the only CERR function revised during the course of this work. The Linux version of the Gamma Plan software package has an available DICOM RT export package. CERR is capable of importing DICOM RT however testing of this method of importation found that several portion of the planC cell array necessary for running the TOGA code were omitted. Thus a MatLab function named RevisePlanC was developed to add the necessary data to the planC cell array to run the TOGA code. Thus the code produced in this work is compatible with either operating system version of Gamma Plan.

Figure 21 displays a treatment plan imported into the CERR format and displayed in the CERR graphical user interface. In figure 21, a pituitary tumor region of interest is outlined in light blue, the right optic nerve region of interest is outlined in orange, and the

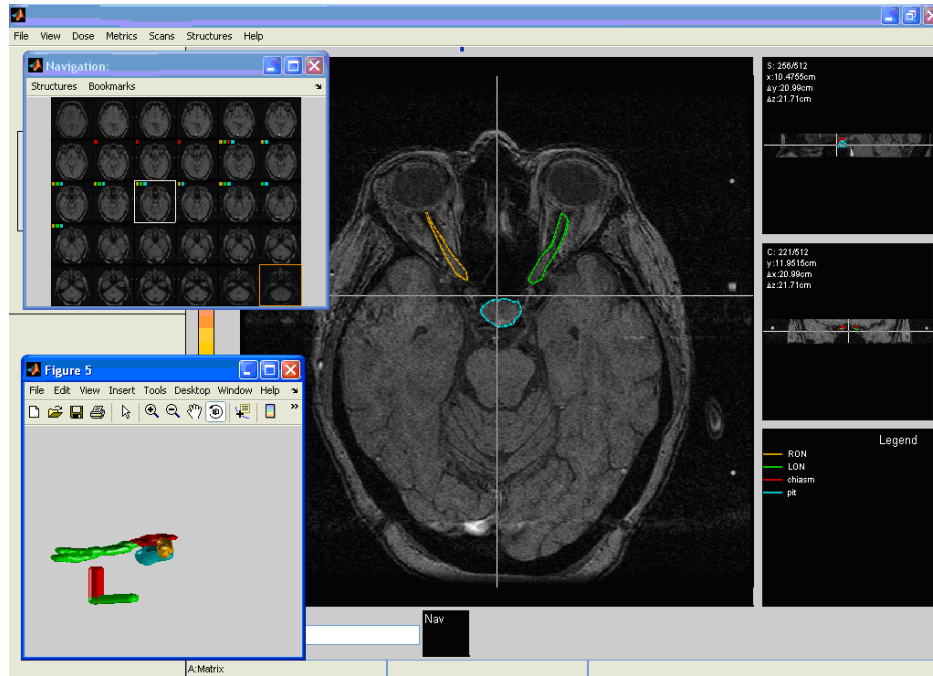


Figure 21 CERR importation of Gamma Knife treatment plan

left optic nerve region of interest is outlined in green. During Gamma Knife treatment planning, a plan would be developed to conform the prescription isodose treatment volume as closely as possible to the pituitary tumor region of interest while limiting the maximum radiation dose incident on the optic nerve regions of interest to as low as possible. Treatment planning statistics are calculated by extracting the value of the radiation dose distribution encompassed by the regions of interest. These treatment planning statistics are utilized to determine the quality of the developed treatment plan.

3.2 Historical Treatment Plan Database

This work uses an auto-associative kernel regression model to map a set of input variables that describe the current target volume to several historical target volumes in the database. The treatment plans that correspond to the selected historical target

volumes are then applied to the current target volume in hope that what was successful in treating similar volumes in the past will be effective in treating the current volume. To test the ability of an auto-associative kernel regression model to perform this task, and to illustrate the method with a set of two dimensional shapes that are easier to visualize, an investigative problem was developed.

3.2.1 Kernel Regression Model for 2D Investigative Problem

The investigative problem features a database of simple shapes made up of 2D circles, 2D ellipsoids, and combination of 2D circles and ellipsoids. Parameters were calculated for the shapes in an attempt to define the shapes mathematically. These parameters included the area of the shape, the square root of the first and second principal component lengths of the shape, and the standard deviation of the distances from the center of mass (COM) of the shape to each point that defines the exterior boundary of the shape. These parameters were chosen to mathematically represent the amount of space the shape contains (area), the dimensions of the shape (principal components), and the irregularity of the shape (COM distance deviation) and are analogous to the parameters utilized in the historical treatment plan database regression model. Figure 22 displays the first shape contained in the database. This shape is a circle with a radius of one and the center of the circle at coordinates $X = 0$, $Y = 0$. All of the shapes contained in the investigative database have arbitrary units and are on the same scale, thus units will not be included in the description of the shape. The area of the shape is 3.14, the square root of the first and second principal component lengths are both one. The standard deviation of the distances from the center of mass to the boundary point is zero. The red lines in the figure display the length and direction of the square root of the first principal component length. The

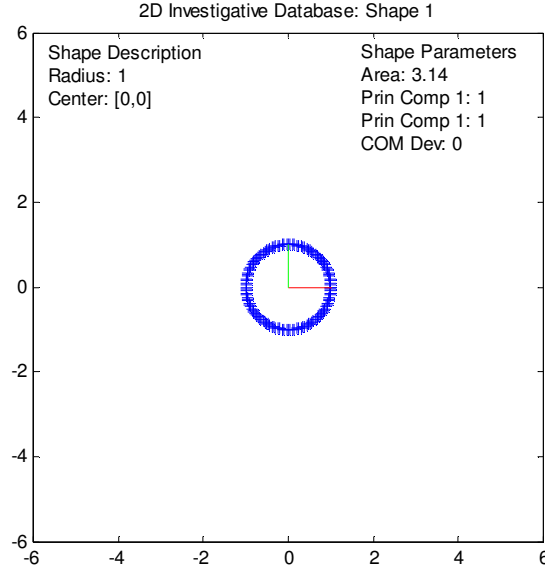


Figure 22 Investigation Database Shape 1

green lines in the figure display the length and direction of the square root of the second principal component length. A total of ten simple shapes are contained in the investigative database. A figure displaying each of the remaining nine shapes can be viewed in appendix B figures 54 – 62. Table 1 displays the parameters extracted from each shape in the database.

An auto-associative kernel regression model was developed to select the shape most similar to an input shape. The selection was performed by computing the Euclidean distance between the discussed parameters of the input shape and each entry in the database. As previously discussed, for an input shape I the distance between each of the parameters and the exemplars matrix is $d(X_D, x_I) = \sqrt{(X_{Data,1} - x_{I,1})^2 + \dots + (X_{Data,n} - x_{I,n})^2}$.

Table 1 Investigative Database Parameters

Shape	Principal Comp.	Principal Comp.	Area	COM distance
1	1	1	3.1416	0
2	3	1	9.4248	0.6968
3	3	2	18.8496	0.3526
4	6	1	18.8496	1.702
5	1.6737	1.0848	5.8299	0.3446
6	3.5721	2.6675	30.8661	0.6608
7	3.7608	2.9858	35.4567	0.5211
8	5	5	78.5398	0
9	1.2182	0.9296	3.5762	0.1528
10	3.6026	2.5524	29.3951	0.6521

The computed distances are then converted to weights using a Gaussian kernel function.

The equation for the Gaussian kernel $w = k_h(d) = \text{Similarity} = \frac{1}{\sqrt{2\pi}h^2} e^{-d^2/(2h^2)}$. The

kernel width h is set to one third of the standard deviation of the parameter. After the Gaussian kernel converts the calculated distances to similarities, the similarities are summed and ranked. The exemplar with the largest sum of similarity is the database shape most similar to the input shape.

Several test input shapes were produced to assess the ability of the kernel regression model to correctly select the most similar shape contained in the database. The first test shape was a circle with radius 1.25 and the center of the circle at coordinates $X = 2$, $Y = 2$. The first test figure was purposely designed to be similar to the first shape contained in the database as seen in figure 22. Figure 23 displays the first input test shape for comparison to the database. The area of the shape is 4.91, the square root of the first and

second principal component lengths are both 1.25. The standard deviation of the distances from the center of mass to the boundary point is zero. The test shape was compared to the database to select the most similar shape. The Euclidean distance between the input shape parameters and each exemplar in the database were computed, converted to similarities, and summed. The result of the comparison, sorted by similarity can be seen in table 2. As expected, the shape with the highest similarity value was shape 1. Figure 24 displays the input test figure and database shape 1 plotted together with the center of masses of the input test figure shifted to match the center of mass of the database shape 1.

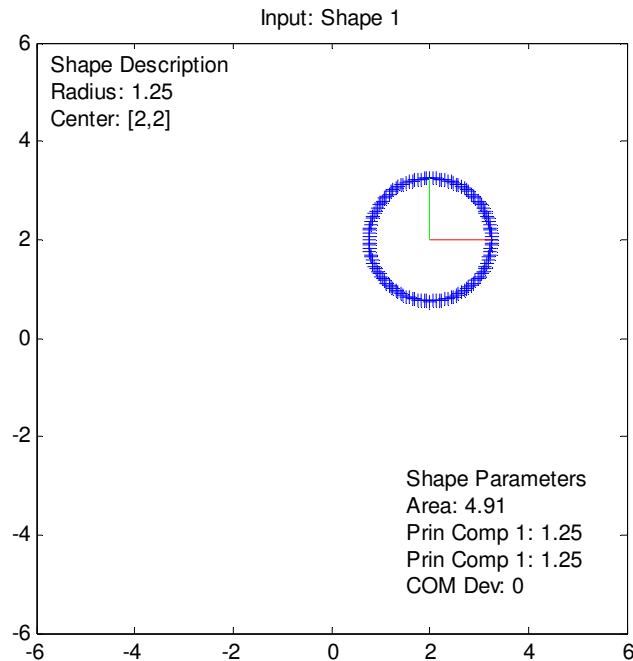


Figure 23 Investigation Problem Input Shape 1

Table 2 Test shape 1 similarity for each database shape

Shape	1	9	5	2	4	8	3	10	6	7
Similarity	3.719	3.401	2.765	1.692	1.034	1.000	0.522	0.018	0.009	0.007

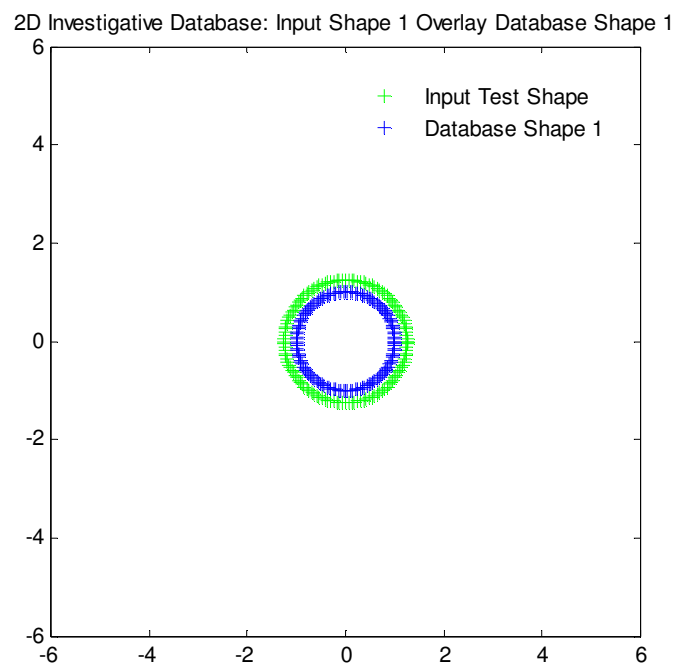


Figure 24 Input Test Shape 1 Overlaid on Database Shape 1

A second more complicated test input shape was produced to further assess the ability of the kernel regression model to correctly select the most similar shape contained in the database. The test shape was a combination of two ellipsoids. The first ellipsoid was centered at coordinate $X = -1$, $Y = 0$ with radius one of 1 and radius two of 2. The second ellipsoid was centered at coordinate $X = 2$, $Y = 0$ with radius one of 3 and radius two of 3. Figure 25 displays the second input test shape for comparison to the database. The area of the second input shape is 23.67, the square root of the first and second principal component lengths are 3.60 and 2.12 respectively. The standard deviation of the distances from the center of mass to the boundary point is 0.59. The test shape was compared to the database to select the most similar shape. The Euclidean distance between the input shape parameter and each exemplar in the database were computed, converted to similarity, and summed. The result of the comparison, sorted by similarity can be seen in the table 3. The shape with the highest similarity value was shape 10.

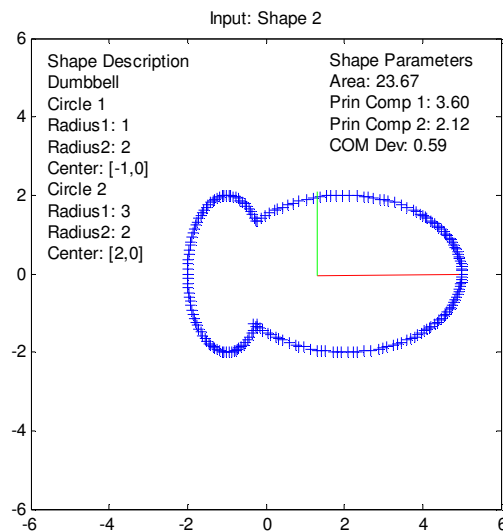


Figure 25 Investigation Problem Input Shape 1

Table 3 Test shape 2 similarity for each database shape

Shape	10	6	3	7	2	4	5	9	1	8
Similarity	3.3041	3.0115	2.6575	2.318	1.5417	0.8542	0.4558	0.0842	0.0651	0.0332

Figure 26 displays the input test figure and database shape 10 plotted together with the center of masses of the input test figure shifted to match the center of mass of the database shape 1 and the input test shape rotated 20 degrees to match the rotation of the database shape. Review of the alignment of the test input shape 2 overlaid on the selected database shape 10 indicates that the two shapes are similar. This demonstrates that the kernel regression model should be capable of comparing objects for similarity and that the method should prove useful in the implementation of the historical database initialization module.

3.2.2 Kernel Regression Model of Historical Plan Database

A database of historical Gamma Knife treatment plans was developed for use in constructing initial populations for the genetic algorithm. The treatment volume parameters contained in the historical plan database are 1) the volume of the treatment region, 2) the principal components lengths of the treatment volume, and 3) an irregularity parameter. The volume parameter is simply the calculated volume of the target region with units of cubic centimeters. An illustration of the principal component lengths are displayed in figure 27. Principal component analysis is a method used to reduce the dimensionality of a block of data without losing a significant amount of information (variability) [69]. The method for computing the principal components will

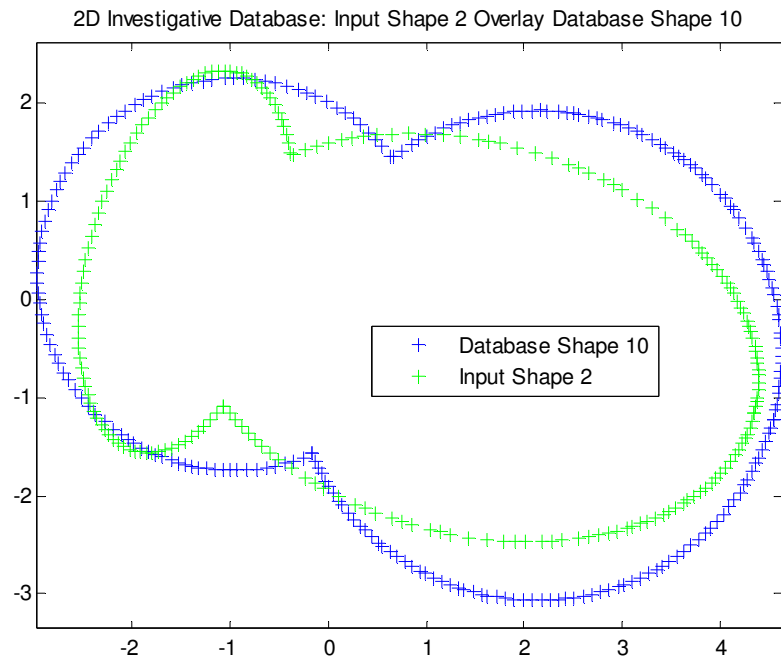


Figure 26 Input Test Shape 2 Overlaid on Database Shape 10

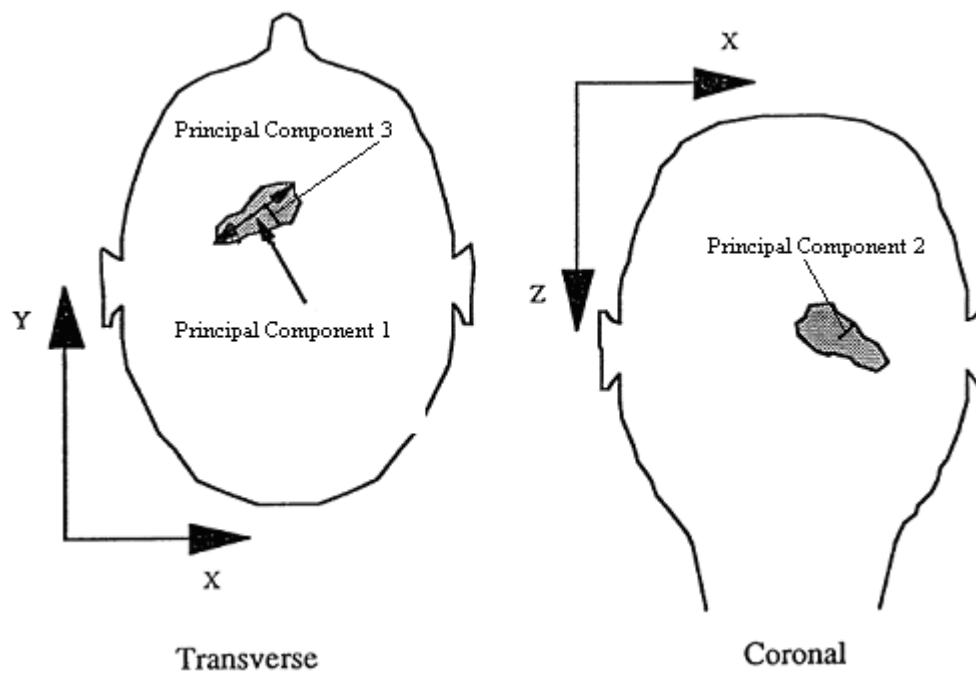


Figure 27 Principal Components

not be discussed here. In this application the dimensionality of the data is not reduced, rather the principal components analysis is used to determine the maximum length of the target volume in the principal component directions. This enables treatment volume comparison regardless of the rotational orientation of the target volumes. The principal components are computed by determining the linear dimension about which the data has the largest variability. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. Each principal component is orthogonal to the other principal components. Calculating the principal component parameters will allow the three dimensional volume to be quantified as a vector. This vector is then compared to the historical plan database to select past similar treatment volumes.

The final variable is an irregularity measure that is equal to the variance of the distance from the volume center of mass to each point that defines the boundary of the volume. This variable is calculated by locating the center of mass of the treatment area, measuring the distance from the center of mass to the exterior surface of the treatment volume in many directions and then determines the standard deviation of the measures. For a perfectly symmetrical spherical volume the parameter would have a value of zero. As the target volume became more irregular in shape the parameters value will increase.

Figure 28 displays a three dimensional plot of a target volume contained in the historical plan database. The first, second, and third principal components of the volume are represented as a red, green, and magenta lines and have values of 2.09 cm, 1.83 cm, and

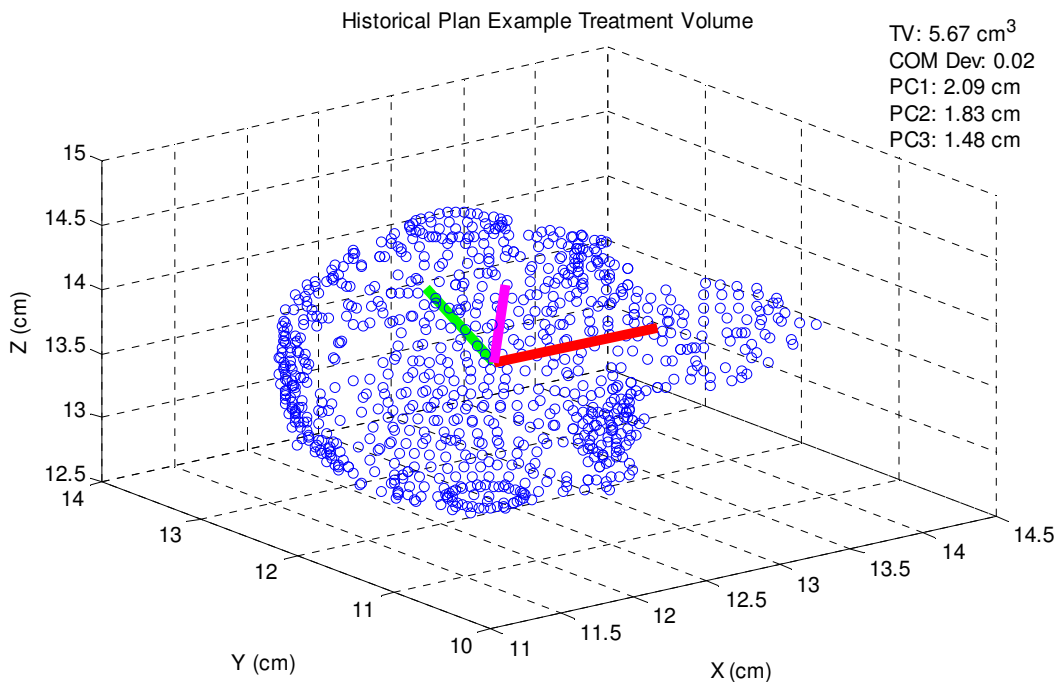


Figure 28 Historical Plan Example Treatment Volume

1.48 cm respectively. The volume of the structure is 5.67 cm³ and the irregularity parameter was 0.02. The described variables are stored in a relational database constructed as a .mat file in MatLab. The structure of the relational database can be seen in figure 29.

The database features treatment target volumes and their corresponding plans performed at the Baptist Regional Gamma Knife (BRGK) in Knoxville Tennessee from January 2005 to December 2008. All patient data is ammonized and protected per the University of Tennessee Institutional Review Board exemption application (Appendix C). All target volumes were delineated by physicians and the plans were constructed by physicists from the center. The database contains 153 target volumes and treatment plans

Historical Plan Database Structure

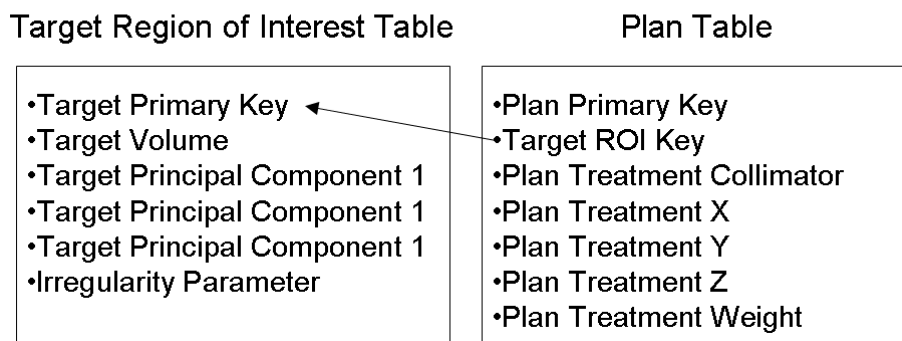


Figure 29 Historical Plan Database Structure

and both relational tables contained in the database can be seen in appendix tables 11 and 12. Patient treatment plans were included in the database if they featured more than one treatment isocenter and a target volume was defined. Table 4 contains the mean and range of the database treatment volume features and Appendix figures 63 – 67 are histograms of the database treatment volume features.

The 153 treatment plans contained in the database are comprised of a total of 1617 treatment isocenters for an average of 10.57 treatment isocenters per target volume. Of the 1617 treatment isocenters; 367 were 4 mm collimators (2.4 per target volume), 1051 were 8 mm collimators (6.9 per target volume), 126 were 14 mm collimators (0.8 per target volume), and 76 were 18 mm collimators (0.50 per target volume). The average weight for treatment isocenters was 0.72 and a histogram of the isocenter weight distribution can be seen in appendix figure 68.

Table 4 Historical Treatment Volume Features

	Mean	Range
Volume (cm ³)	4.96	0.1 – 27.8
Principal Component 1 (cm)	2.17	0.75 – 6.25
Principal Component 2 (cm)	1.63	0.47 – 3.32
Principal Component 3 (cm)	1.22	0.14 – 2.54
Irregularity Parameter	0.0803	0.0034 – 1.2

The historical database is a relational database between the table containing the target region of interest information and the plan treatment isocenter information as shown in figure 29. An individual treatment isocenter contained in the database is related to a target volume by the target region of interest (ROI) key variable. All treatment isocenters that belong to a particular treatment plan have the same value for the ROI key variable which corresponds to a particular target volume. If during the kernel regression process a target volume is selected, all treatment isocenters belonging to the corresponding treatment plan are extracted and applied to the target to be optimized. This is accomplished by extracting all treatment isocenters with the corresponding ROI key.

As with the 2D investigative problem, an auto-associative kernel regression model was developed to select a set of historical treatment volume that are similar in size and shape to the current target volume. The selection was performed by computing the Euclidean distance between the discussed treatment volume parameters and each entry in the database. For an input target volume, I the distance between each of the parameters and

the exemplars contained in the database D is

$$d(X_D, x_I) = \sqrt{(X_{Data,1} - x_{I,1})^2 + \dots + (X_{Data,n} - x_{I,n})^2} .$$

The computed distances are then

converted to weights using a Gaussian kernel function with equation

$$w = k_h(d) = \text{Similarity} = \frac{1}{\sqrt{2\pi}h^2} e^{-d^2/(2h^2)} .$$

The kernel width h is set to one third of the

standard deviation of the parameter. After the conversion the similarities are summed

and ranked. The exemplars with the largest sum of similarities are the historical

treatment volumes most similar to the current target volume. The developed computer

code allows the user to specify the number of historical treatment plans to extract for

inclusion in the GA initial population. This is performed by extracting the treatment

plans that correspond to the historical target volumes with the highest ranked similarities

until the number of specified plans has been selected.

After the historical treatment plans have been extracted for inclusion in GA initial

population, it is necessary to perform a rigid linear transformation of the plan onto the

current target volume. This is necessary to account for positional and rotational

orientation differences between the current target volume and the selected historical

target volumes. The extracted treatment plans are what is actually transformed during

this process. An initial linear transformation is performed that shifts the treatment plan

by the difference of the average X, Y, and Z coordinates of the treatment plan and the

center of mass of the current target volume. This shift roughly positions the treatment

plan on the target volume. A set of rigid rotational and translational transformation

$$\text{matrices of the form } R_x(X, \alpha) = \begin{bmatrix} X & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}, R_y(Y, \beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & Y & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$\text{, and } R_z(Z, \gamma) = \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & Z \end{bmatrix} \text{ are then applied to the treatment plan to fine tune}$$

the alignment of the plan and the target volume [(Goldstein 1980, pp. 146-147 and 608; Arfken 1985, pp. 199-200)]. A genetic algorithm optimization is employed to compute the optimal values of X, Y, Z, α , β , and γ parameters in the transformation matrix. The genetic algorithm is set to optimize the six parameters for ten generation with a fifteen member population. The cost function for the optimization is the Paddick conformity index presented in detail in section 3.5. The Paddick conformity index is utilized because it contains elements of target coverage and normal tissue sparing resulting in the rigid transformation being as clinically relevant as possible. During the genetic algorithm optimization the values of the positional transformation parameters (X, Y, & Z) are allowed to vary between -2 and 2 centimeters. The rotational parameters are allowed to vary between 0 and 2π . This allows the genetic algorithm to rotate the treatment plan across any dimension.

Figure 30 displays an example of a single cut of a treatment plan isodose distribution mirrored across the Y plane. This is accomplished with the following transformation parameters: $X=Y=Z=0$, $\alpha=0$, $\beta=\pi$, $\gamma=0$. The ability of the rigid transformation to rotate the treatment plan is of particular use when aligning acoustic schwannoma historical treatment plans. Acoustic schwannomas are benign primary intracranial tumor of the

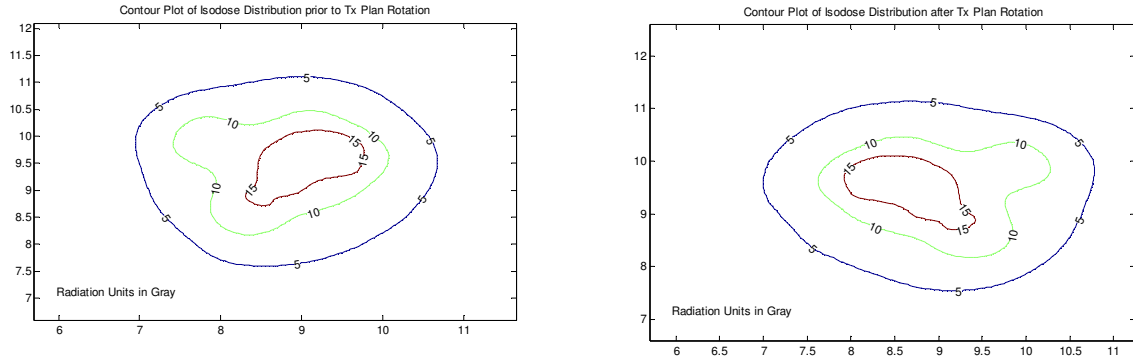


Figure 30 Isodose Distribution Rotation about Y axis

myelin-forming cells of the vestibulocochlear nerve, the seventh cranial nerve. As seen in figure 28, these tumors have a general shape of a medially located central spherical volume within the internal auditory canal and an extension laterally into the posterior fossa to occupy the angle between the cerebellum and the pons (cerebellopontine angle). Due to the described anatomy, left and right acoustic schwannomas generally have a somewhat mirrored shape. The ability of the rigid transformation to rotate the treatment plan will allow historical treatment plans for left sided acoustic schwannoma to be applied to right sided tumors, and vice versa.

3.3 Initial Population by Geometric Optimization

In addition to the historical plan initiation module, a geometric initiation module was developed to provide treatment plans for the genetic algorithm initial population. The geometric optimization module functions by iteratively adding single treatment isocenters on the target volume. The position of a single isocenter is optimized to maximize the geometric coverage of the target volume. The target volume is represented by a binary three dimensional matrix of the same dimension as a dose calculation matrix. Elements

of the matrix are assigned a value of one if they are encompassed by the geometric boundary of the target volume and zero if they are not. A genetic algorithm is then utilized to iteratively determine the optimal placement of each isocenter on the target volume matrix so that the maximum number of non-zero elements is covered. The objective function for the optimization is the percent reduction of the number of non-zero elements of the target volume matrix. The module requires the number of treatment isocenters, the maximum collimator size to be used, and a coverage parameter that determines the amount of space each isocenter covers be specified. The coverage parameters value is the percent isodose volume of a single treatment isocenter and the value can vary between 0.4 and 0.9. Table 5 contains the percent isodose volume of each of the four collimators sizes versus the coverage parameter. The initialization module computes the collimator size to place on the target volume matrix. This is performed by comparing the volume of the uncovered area in the target volume matrix to the table of percent isocenter volume for the specified coverage parameter. The largest collimator size is select whose volume does not exceed the remaining volume of the uncovered target for the specified coverage parameter. The position of the selected isocenter is then optimized by a genetic algorithm for the maximum reduction of non- zero elements of the

Table 5 Percent Isodose Volume cm³ for each collimator by coverage parameter

Collimator	Coverage Parameter					
	0.4	0.5	0.6	0.7	0.8	0.9
4 mm	0.1244	0.0894	0.066	0.0479	0.0318	0.0135
8 mm	0.7988	0.5885	0.4534	0.3538	0.2625	0.1461
14 mm	3.9	2.9	2.3	1.9	1.6	1.2
18 mm	7.7	5.8	4.7	4	3.4	2.7

target. To determine if a non-zero element is covered for a specified isocenter position, a rudimentary coverage calculation is performed where a matrix representing the isodose distribution of the single isocenter is added to dose matrix of zeros with the same dimension as the target volume matrix. The non-zero elements of the dose calculation matrix are located and the elements of the target volume that correspond to the same position are set to zero. The result is a conversion of a portion of the target volume from ones to zeros representing the coverage of the single isocenter. Figure 31 illustrates the geometric optimization process.

3.4 Dose Calculation

The calculation of radiation dose distribution in Gamma Plan was discussed in the literature survey. The calculation of the dose at a single point requires the tangential

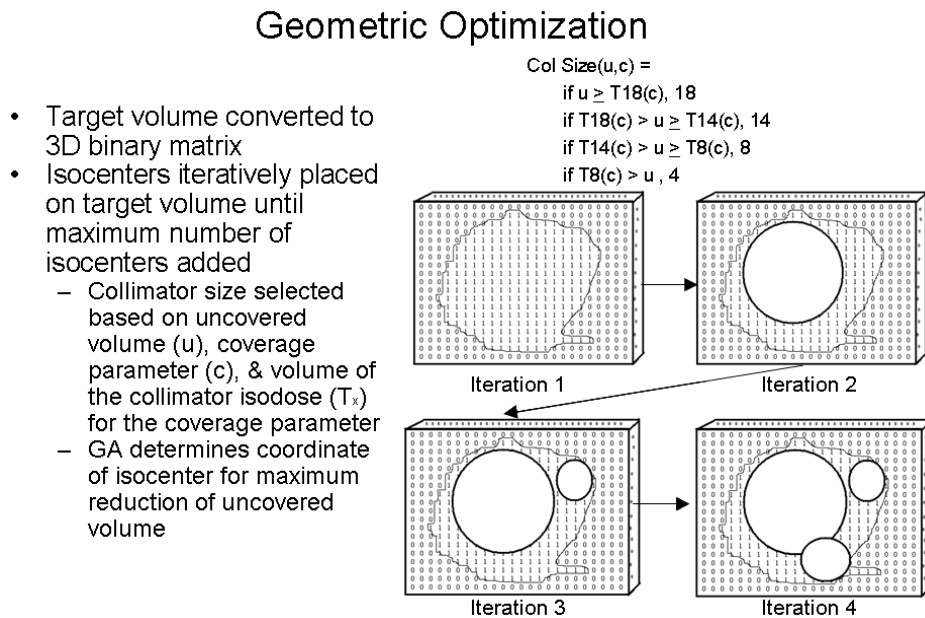


Figure 31 Geometric Optimization

distance of the point from the axis of the single Cobalt-60 beam and the tissue depth at the tangential point along the beam axis to be calculated and substituted into an algebraic equation including an exponential factor summed over all 201 Cobalt-60 radiation sources. The calculation would then be repeated for each dose calculation point. This process is somewhat computation intensive. The optimization process developed in this work requires the radiation dose distribution from multiple plans to be calculated at each optimization iteration. Considering the number of necessary calculations a rapid robust dose calculation is essential to avoid long optimization time.

Kjäll noted that the Gamma Knife radiation distribution is fairly independent of asymmetric depth [61]. This lack of depth influence on the radiation dose distribution can be seen in figure 32 which displays a film acquisition of the radiation distribution from the Gamma Knife collimator near the edge of a phantom [61]. The lack of change in the distribution can be contributed to the large number of Cobalt-60 beams utilized by the Gamma Knife. While several beams may pass through only a small amount of tissue, the majority of the beams pass through a similar amount of tissue before the intersection at the isocenter. This results in the Gamma Knife radiation dose distribution remaining fairly unchanged regardless of its location in the cranium. In this work a rapid radiation dose calculation will be performed by simply “summing” standard dose matrices that represent an individual shot. A standard model of each of the four radiation helmets (4mm, 8mm, 14mm, and 18mm) at four different gamma angles (72°, 90°, 110°, and 125°) will be produced from an 8 centimeter spherical phantom. These standard models will represent individual shots and will be summed over a radiation dose calculation grid

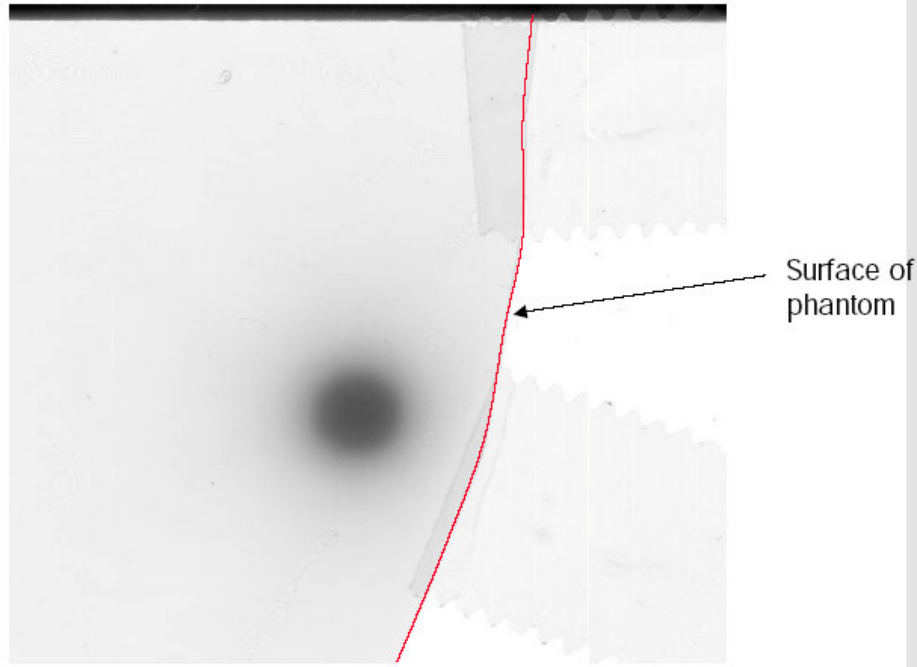


Figure 32 Gamma Knife radiation distribution near edge of phantom

according to the planned shot location. The dose distribution will then be normalized to the maximum value to produce percent isodose lines to evaluate conformance objectives.

This simple calculation model greatly improves computation time enabling the optimization of the treatment plans in a reasonable amount of time. Thus, the equation

for the normalized rapid radiation dose calculation is $D_r(x, y, z) = \text{norm} \left(\sum_{i=1}^n D_i(x, y, z, h) \right)$.

Figure 33 and appendix figures 69 and 70 display a dose line profile comparison in the X,

Y, and Z plane respectively of the dose distribution calculated by the Gamma Plan

treatment planning software and the rapid radiation dose calculation developed for this

work. The calculation is for a twelve isocenter treatment plan featuring two 14 mm

collimator and ten 8 mm collimator isocenters. In the figure the red line represents the

Gamma Plan calculated dose distribution and the orange line represents the rapid

radiation dose calculation. As evident from the figures the rapid radiation dose calculation closely matches the Gamma Plan calculation and should be an effective method of determining the dose distribution for the optimization. The rapid radiation dose calculation was developed as an original contribution of this dissertation and the computer code for the calculation can be seen in the Appendix: MatLab Code.

To further assess the accuracy of the rapid radiation dose calculation a two dimensional gamma analysis was performed for the transverse, coronal and sagittal planes. A gamma

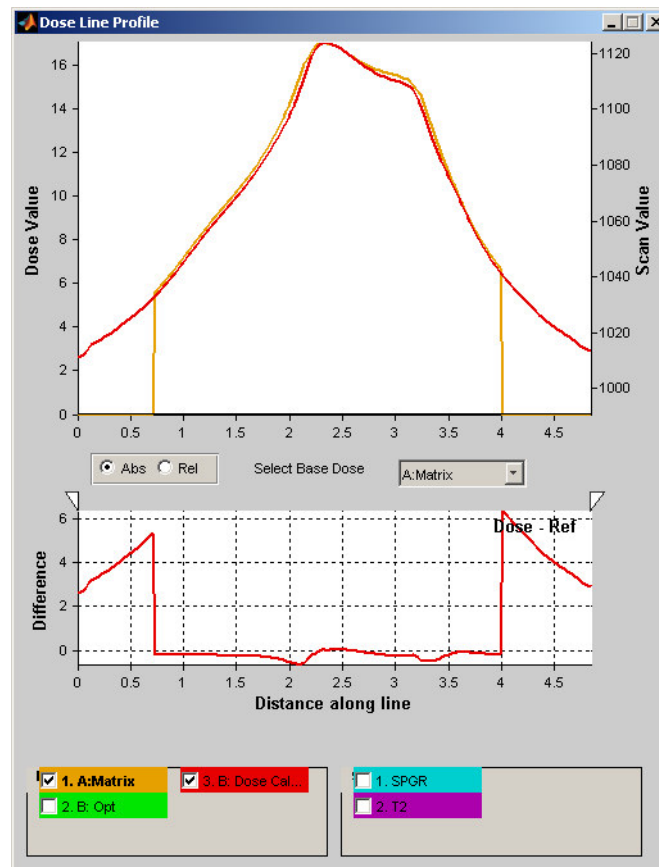


Figure 33 Gamma Plan versus Rapid Dose Calculation Comparison X Plane

analysis is a comparison between two isodose distributions that accounts for both the percent difference between the distributions and the distance to point agreement of the two distributions [68]. The equation for the gamma index calculated between a reference dose distribution 1 and dose distribution 2 is $\gamma(x_1, y_1) = \min_{x_2, y_2} \{\Gamma_1(x_2, y_2, D_2)\}$ where

$$\Gamma_1(x_2, y_2, D_2) = \sqrt{\frac{\Delta r^2}{\Delta r_{\max}^2} + \frac{\Delta D^2}{\Delta D_{\max}^2}}, \Delta r = |\vec{r}_2 - \vec{r}_1| = (x_2 - x_1)^2 + (y_2 - y_1)^2, \&$$

$\Delta D = D_2(\vec{r}_2) - D_1(\vec{r}_1)$. The gamma index is calculated for each dose point in the reference dose distribution 1, by comparing the dose at every calculation point and the distance to each of these calculation points in distribution 2 by some maximum acceptable dose difference $\Delta r_{\max}$ and distance difference $\Delta D_{\max}$ to determine $\Gamma_1(x_2, y_2, D_2)$. The minimum value for the set of $\Gamma_1(x_2, y_2, D_2)$ is then calculated to determine $\gamma(x_1, y_1)$. For this comparison $\Delta r_{\max}$ is set to 3 millimeters and $\Delta D_{\max}$ is set to 3%. Traditionally if the fraction of gamma index with values less than 1 is greater than 95%, the two dose distributions are considered in good agreement. Figure 34 and appendix figures 71 and 72 displays the gamma comparison between the Gamma Plan and rapid radiation dose calculation for the previously described comparison in the transverse, coronal and sagittal planes respectively. The fraction of calculation points with gammas less than one is 97.5%, 95.84%, & 96.57% for the transverse, coronal and sagittal distributions indicating that the rapid dose distribution is in good agreement with the reference Gamma Plan radiation dose calculation. Thus, the rapid radiation dose calculation should be able to be utilized to quickly determine the radiation dose distribution during the genetic algorithm dose optimization process.

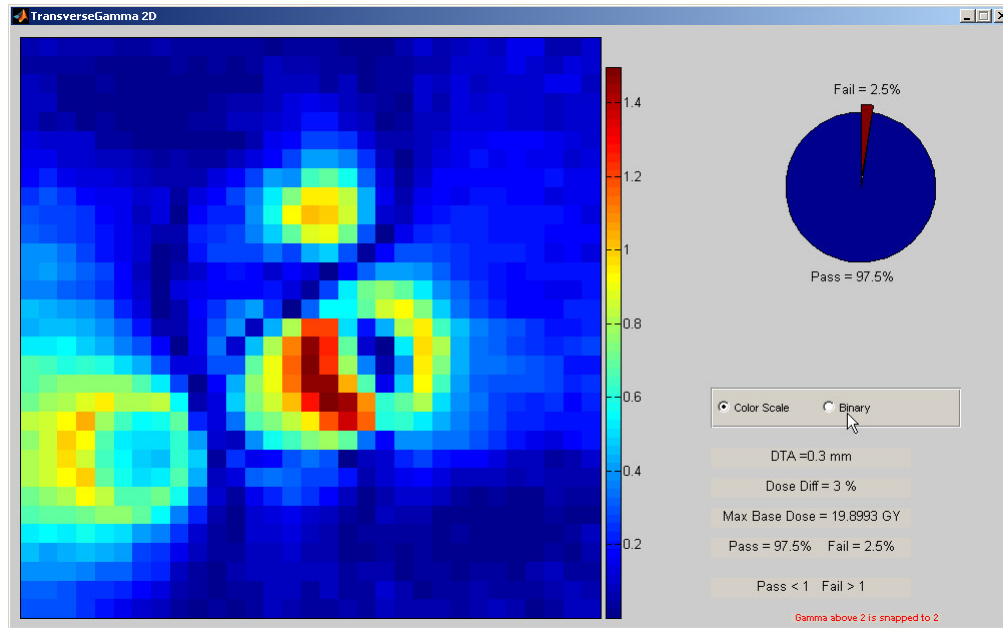


Figure 34 Gamma Plan versus Rapid Dose Gamma Distribution Transverse Plane

3.5 Optimization Parameters & Fitness Function

Generally, the objective of Gamma Knife treatment planning is to cover the treatment site with the prescribed isodose line (maximize coverage), while maximizing conformity (minimize dose to normal tissue) and not exceeding the treatment tolerances of any critical structures in the vicinity of the treatment site. Several criteria are used to quantify these objectives. These criteria are included in a multi-objective optimization fitness function by giving each optimization parameter a user defined weight included in the cost function evaluation. Thus the fitness function for the optimization is a weighted average of several conformity and plan quality indices.

Paddick [62] describes a conformity index that is the ratio of treatment site coverage to

normal tissue sparing such that $Paddick\ Index = \frac{TV_{PIV}}{TV} \frac{TV_{PIV}}{PIV} = \frac{TV_{PIV}^2}{TV \times PIV}$ where

$TV_{PIV} = Volume\ of\ target\ covered\ by\ Rx\ Isodose$, $TV = Volume\ of\ target$, and

$PIV = Percentage\ Isodose\ Volume$. This conformity index varies from zero to one with

one being a perfect plan. As the genetic algorithm optimization is a minimization the

value of the conformity index is multiplied by negative one and then added to one so that

the value varies between zero and one with zero being an optimal plan.

The Paddick conformity index accounts for treatment site coverage and conformity,

however in the treatment of certain disease, such as metastasis, it is more important to

have adequate coverage than to be extremely conformal due to the presence of unseen

microscopic disease on the periphery of the treatment site. Thus, it may be useful to

optimize a treatment plan based on percent isodose coverage of the treatment site.

Percent isodose coverage is computed by extracting the calculated dose at each point

encompassed by the target volume. The percentage of these points whose dose value

exceeds the prescription is the percent isodose coverage. Adequate coverage is generally

considered between 95-100% with 100% prescription isodose coverage considered

optimal. Again as the genetic algorithm optimization is a minimization the percent

isodose coverage is multiplied by negative one and then added to one so that the value

varies between zero and one with zero being an optimal plan.

The Radiation Therapy Oncology Group (RTOG) conformity index is one of the most

commonly reported conformity indices and is included as an available portion of the cost

function. The index is defined as the ratio of volume of the prescription isodose to the target volume [10]. The index has an optimal value of 1. Index values between 0.9 to 1 and 1 to 2 are considered acceptable, 2 to 2.5 are minor deviations, and less than 0.9 and greater than 2.5 or greater are considered major deviations from an acceptable plan. A deficiency with the RTOG conformity index is the index contains no information about target coverage. A developed treatment can have an optimal RTOG conformity index value and not adequately cover the target volume if the radiation distribution is not centered on the target volume. Thus the RTOG conformity index should be given limited weight during the calculation of the composite cost function. To place the RTOG conformity index on the same value scale as the other conformity indices, one is subtracted from the value of the RTOG conformity index such that the index has a value of zero for an optimal plan. Again as the RTOG conformity index does not contain any information regarding target volume coverage it must be used in conjunction with the Paddick conformity index or the percent isodose coverage to effectively optimized the treatment plan.

The Radiation Therapy Oncology Group (RTOG) coverage index is also included as an available portion of the cost function calculation for the genetic algorithm optimization. The index is defined as the ratio of the minimum dose target volume divided by the prescription isodose [10]. The index has an optimal value of 1. Index values less than 0.9 are minor deviations, and less than 0.8 are considered major deviations. The RTOG coverage index contains no information about the conformity of the treatment plan. The index has an optimal value provided the prescription isodose covers the entire volume. A

plan could have an optimal value for the coverage index but lack adequate conformity and treatment too much normal tissue. Thus the RTOG coverage index must be used in conjunction with another conformity index that contains some conformity component. To place the RTOG conformity index on the same scale as the other conformity indices the value of the RTOG coverage index is multiplied by negative one and then added to one so that the value varies between zero and one with zero being an optimal plan.

The Gradient Index is included as an available portion of the cost function calculation for the genetic algorithm optimization [63]. The index is meant to compliment the Paddick conformity index. It is defined by the ratio of the volume of one half the prescription isodose to the prescription isodose volume. The index represents how rapidly the dose distribution decreases outside of the target volume. The thinking behind the index is that a plan may have adequate conformity and coverage of the prescription isodose but lack the rapid fall off of dose that is characteristic of Gamma Knife planning. This type of phenomenon can be caused by placing inappropriately large collimator isocenter on the boundary of the target volume. This type of planning can result in good conformality of the prescription isodose but inadequate fall off of the lower isodose levels. However, the index contains no information regarding prescription coverage of the target volume thus it must be used in conjunction with an index that contains some information regarding target coverage. The index has an optimal value of two with a value less than three being acceptable. To place the gradient index on the same scale as the other conformity index two is subtracted from the value so that the value of an optimal plan is zero.

When the treatment site is in the vicinity of a critical structure, such as optic structures, it is crucial not to exceed the radiation dose tolerance of the structure. Thus, it is useful to optimize the radiation dose delivered to the critical structures to ensure it is below tolerance. This dose criterion is coded as a sum of square error constraint in the optimization. During the evaluation of the cost function value for the critical structure tolerance objective, the value of the dose calculation points encompassed by the structure are extracted and compared to a user provided tolerance. If all extracted point values are less than the provided tolerance the critical structure cost function value is set to zero. If any point values exceed the tolerance the cost function is calculated as a sum of square error where $Cost_{CS} = \sum_i^n (D_i - T)^2 / n$ where D_i is set of doses to the critical structure that exceed the provided dose tolerance T and n is the number of calculation points that exceed the tolerance.

A weighting vector is utilized to produce a composite cost function value from each of the components. Each of the calculated conformity indices are multiplied by an element of the weight vector that dictates the importance of that index towards the optimization objective. The resulting values are then summed to produce a composite cost value for the cost function. The weighting vector is supplied by the code user and is specified based on the users experience and the objective of the treatment. For instance, for the treatment of a solitary brain metastasis located in one of the frontal lobes of the brain the user may specify that percent isodose coverage and gradient index are most relevant for the treatment optimization because adequate coverage and rapid dose fall off may most impact the patient's prognosis. Conversely, for the treatment of an acoustic schwannoma

in a hearing intact patient, the user may specify conformity of the prescription isodose and the cochlea structure maximum dose as the indices most relevant to the quality of the treatment plan.

3.6 Genetic Algorithm Optimization

This work uses a genetic algorithm to optimized Gamma Knife three dimensional treatment planning. To test the ability of the genetic algorithm optimization method to perform this task, and to illustrate the method within an easy to visual one dimensional setting, an investigative problem was developed.

3.6.1 Genetic Algorithm fitting of a One-Dimensional Test Problem

As discussed Gamma Knife radiation dose computation is analogous to a normalized summation of Gaussian distributions. A one dimensional test problem was developed to assess the ability of a genetic algorithm optimizer to determine the size and position of a collection of summed Gaussian distributions. This was performed to determine if the genetic algorithm optimizer could effectively be used to optimize a problem similar to the Gamma Knife three-dimensional dose distribution optimization. The test problem was developed within the MatLab Genetic Algorithm and Direct Search Toolbox [64]. The Genetic Algorithm and Direct Search Toolbox is a collection of functions that extend the capabilities of the Optimization Toolbox and the MATLAB numeric computing environment. The toolbox functions can be called from the MatLab command line or through the Genetic Algorithm Tool, a graphical interface to the genetic algorithm. The genetic algorithm tool optimizes the objective function by:

- 1) Creating an initial random population of individuals to optimize,

- 2) Modifies the initial population through a process that emulates natural selection to optimistically produce a population of better performing individuals on the objective function. This is accomplished by:
 - i. Scoring each individuals initial performance on the objective function,
 - ii. Scale the fitness values to a more usable range,
 - iii. Select the parents from the population to populate the next generation,
 - iv. Select elite members of the population to pass on to the next generation without modification, and finally
 - v. Produce children to populate the next generation.
- 3) The optimization is stopped when some set of stopping criteria is met. This may be a time limit, generation limit, fitness limit, or optimization performance stall limit.

The genetic algorithm tool has three main rule types for producing children to populate future generation; Selection rules (selects parents), Crossover Rules (combine parents to form children), and Mutation Rules (applies random changes).

The investigative problem was performed using a test dataset. The dataset was a combination of three Gaussian distributions of the form $f(x, \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$ on the interval $x = -10$ to 10 with a spacing of 0.1 . The Gaussian distributions were centered at $c = -5, 1$, and 6 with σ parameter of $3, 1$, and 2 respectively. The sigma parameter was represented by a two bit binary variable of the form $[0,0] = 1, [0,1] = 2, [1,0]=3$, and $[1,1]=4$. The binary representation of the sigma parameter is used to more closely follow the methodology of the Gamma Knife treatment planning optimization where the collimator size is represented by a two bit binary variable. Finally, each Gaussian

distribution was given a weight of 1, 1.5, and 1.3 respectively to represent the weight parameter for Gamma knife treatment planning. Figure 35 displays the three summed Gaussian distribution plotted on the interval, normalized to the maximum value of the

composite distribution computed by $f_c(x, \sigma, c) = \left(\sum_{i=1}^3 e^{\frac{-(x-c_i)^2}{2\sigma_i^2}} \right) / \max \left(\sum_{i=1}^3 e^{\frac{-(x-c_i)^2}{2\sigma_i^2}} \right)$. A

genetic algorithm optimizer was then utilized to iteratively determine the position (x), weight (w), and Gaussian spread (σ) of the three Gaussian distribution function used to create the test data with no prior knowledge of the values of these parameters. The cost function for the optimization was a sum of square error, $Cost = \sum (f_{opt}(x, \sigma, c) - f_c(x, \sigma, c))^2$, between the test data and the distribution calculated by the optimizer. The parameter

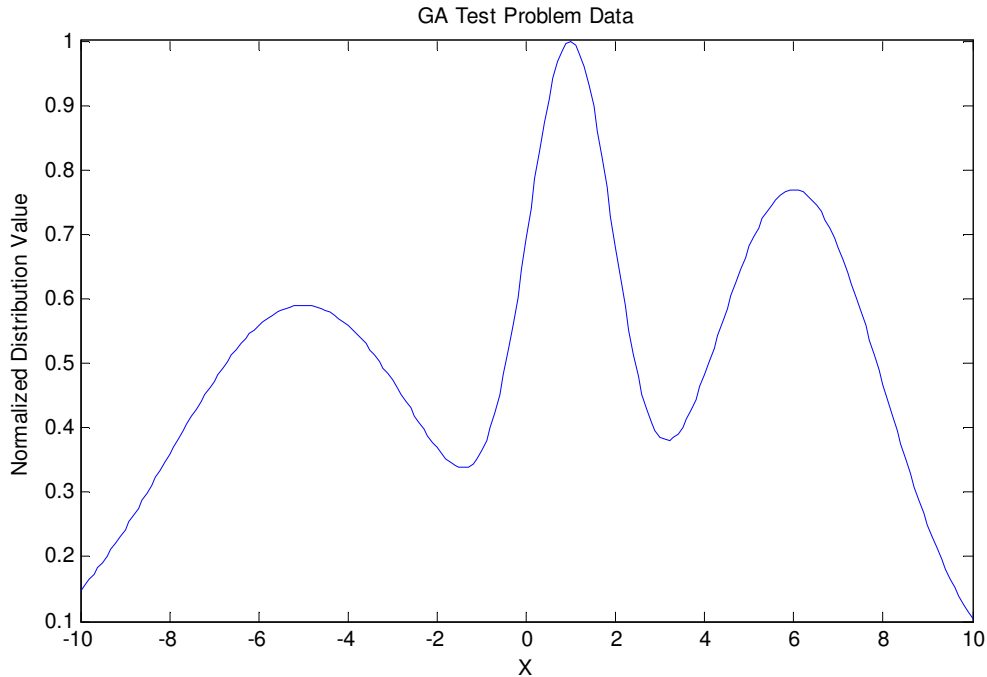


Figure 35 GA Test Problem Data

vector to be optimized featured 12 parameters to represent the three Gaussian distributions. For each distribution the first two parameters were a binary representation of the Gaussian sigma, the third parameter the X coordinate, and the forth parameter the weight of the distribution towards the composite. To account for the optimization of a binary and continuous variable, custom creation, crossover, and mutation functions were created. This required coding the creation function to create binary data parameters to represent the Gaussian sigma parameter and continuous data for the remaining parameters. A uniform crossover function was utilized so that parameters selected from an individual for crossover were simply interchanged. Mutation of the binary variables was achieved by change the binary value for the parameter if the value generated from a random normal distribution exceeded a threshold tolerance. The threshold was set to 0.01 meaning that mutation was common in those individuals selected for mutation. Mutation of the continuous variables was achieved by adding a random normal value multiplied by a scale factor to the parameter value. The scale value was set at 0.5 for the optimization. The genetic algorithm option default values were used for the optimization with a population size of 80 to allow an exhausted search of the problem space, the number of generation allowed for the optimization was set to 500, the allowed range of the X coordinate was -10 to 10, and weight was allowed to range from 0.3 to 1.7 (value not significant as the composite distribution is normalized), and binary representation of the Gaussian σ parameter was allowed to take on all possible values. The default method for the scaling of rank was used and the selection function was a stochastic uniform.

The genetic algorithm was allowed to optimize the parameter values and the results of the optimization can be seen in figure 36. As evident from figure 36, the genetic algorithm optimization was able to closely fit the test data with a final sum of square error of 0.0035. The optimization was terminated after 305 generation because the average change in the fitness value was less than the set tolerance of 10^{-6} . Table 6 contains the final results of the optimization of the parameters versus the parameters used to create the data. The optimization was able to closely match the parameter values used to create the test data. The genetic algorithm optimization appears to be well suited to perform this type of optimization.

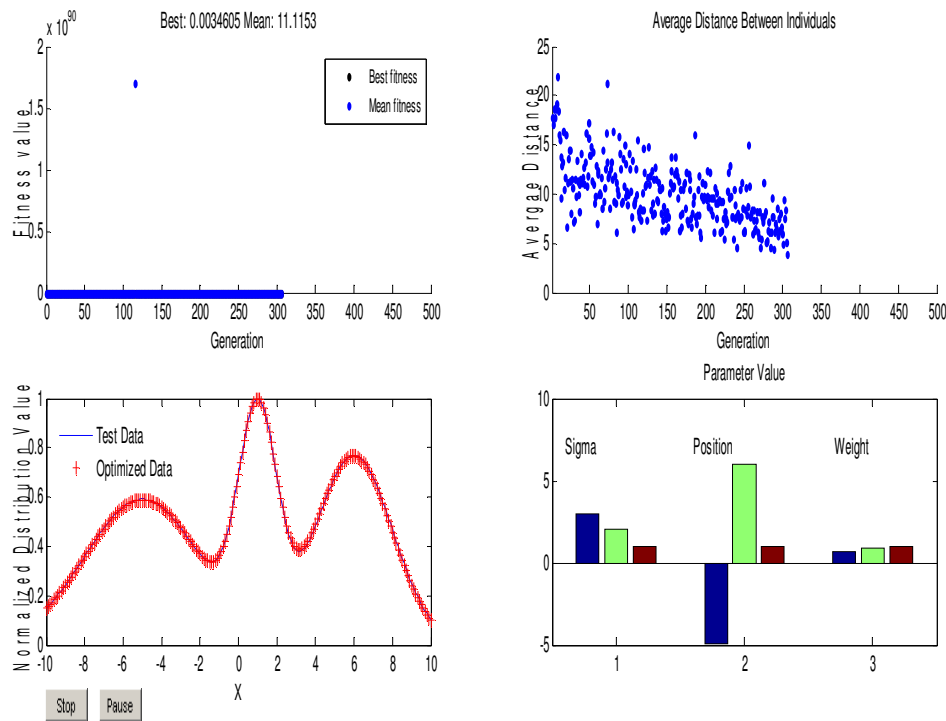


Figure 36 Genetic Algorithm Optimization of Test Problem

Table 6 Genetic Algorithm Optimization of Test Problem Results

	Distribution											
	1				2				3			
	σ_1	σ_2	X	W	σ_1	σ_2	X	W	σ_1	σ_2	X	W
Parameter	0	1	-5	1.0	0	0	1	1.5	1	0	6	1.3
Opt Parameter	0	1	-4.99	0.676 1.0*	0	0	1.004	1.013 1.50*	1	0	5.965	0.878 1.30*

*Optimization weight parameter normalized to maximum value of weight parameter

3.6.2 Genetic Algorithm Optimization of Gamma Knife Treatment Planning

The work optimizes a collection of treatment parameters that comprise an individual member of a genetic algorithm population to arrive at an optimal treatment plan for a target volume. The optimization begins with an initial population representing several treatment plans that is provided to the algorithm by the initialization module. The genetic algorithm may add additional random treatment plans to the population if the number of plans in the population is less than a specified size. The algorithm then creates a sequence of new populations of treatment plans using the plans in the current generation to create the next population of plans in the hope of arriving at an optimal solution. This is performed by: 1) scoring each plan on the objective function, 2) scaling the raw fitness scores to convert them into a more usable range of values, 3) select high performing plans based on their fitness, 4) classifying the highest performing plans as elite and passing them on to the next population, 5) producing children plans from the parents plans through a process of mutation and crossover, and finally 6) replacing the current population with the children to form the next generation. The algorithm continues the process of replacing the population with a subsequent population until a stopping criterion is met. Each individual in the population is represented by a vector of length six

times n , where n is the number of treatment isocenters in the plan. Each treatment isocenter is represented by six parameters that are either binary or continuous. Parameters 1 and 2 are a binary representation of the treatment collimator where $[0,0] = 4$ mm collimator, $[0,1] = 8$ mm collimator, $[1,0] = 14$ mm collimator, and $[1,1] = 18$ mm collimator. Parameters 3, 4, and 5 are continuous and are the X, Y, & Z coordinates of the treatment isocenter. Parameter 6 is continuous and is the weight of the treatment isocenter. The fact that the vector representation of the treatment plan contains both binary and continuous elements requires custom creation, crossover, and mutation functions to be developed. Figure 37 is a graphical user interface of the TOGA code that initializes and runs the collection of MatLab function that performs the genetic algorithm optimization of the target volume.

The TOGA GUI allows the user to specify treatment plan parameters and several optimization parameters. The treatment plan parameters include the structure to be

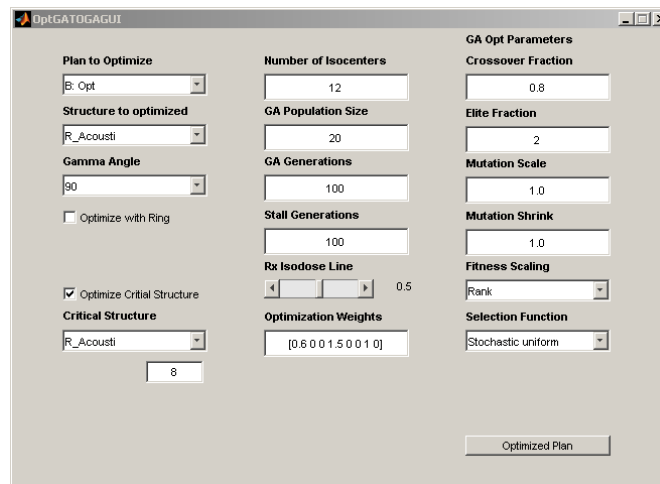


Figure 37 TOGA Genetic Algorithm Optimization GUI

optimized, the gamma angle of the patient's cranium for the optimized treatment, the inclusion and dose tolerance of a critical structure if desired, the number of treatment isocenters, and the prescription isodose line.

- The structure to be optimized is the target volume for the planning objective.
- The gamma angle has previously been discussed and is the angle with respect to the treatment couch of the patients frame attached to the Gamma Knife. This parameter is dependent on the location of the target volume in the patient's cranium. Generally a neutral angle of 90° is used, but if the target volume is biased towards the anterior (gamma angle of 72°) or posterior (gamma angle of 110° , or 125°) portion of the brain an angle may be necessary.
- The inclusion of a critical structure is dependant on if the target volume is in close proximity to some particularly radiosensitive tissue. The cost function for this type of optimization was discussed in section 3.5. The inclusion of a critical structure is not necessary in the optimization but is available if desired.
- The number of treatment isocenters is also specified. This value should be determined from the initialization module and should be set to the maximum number of treatment isocenters of any plan produced by the initialization module.
- The prescription isodose value is also specified in the optimization. Traditionally, in Gamma Knife treatment planning the 50% isodose value is chosen as the prescription line, however the user is able to specify the line as desired.

The optimization parameters that the genetic algorithm optimization GUI allows the user to specify includes the genetic algorithm population size, the number of generations to perform the optimization, the stall generation, the weighting vector for determining the

composite cost function value, the crossover and elite fraction, the mutation scale and shrink, the method by which the individuals of the population are scaled, and the selection function.

- The genetic algorithm population size is the number of individuals to be optimized. The larger the population size the more diverse it is and the more thoroughly the genetic algorithm is able to search the parameter space. However the trade off is computational time, in that as the population size increases optimization time become longer. Experimentation has found that a population size of twenty provides a diverse population for arriving at the optimal solution while not sacrificing the works objective of developing a treatment plan as quickly as possible. A larger genetic algorithm population size can be specified than is provided by the initialization module. In this case, the remaining members of the initial populations are generated with a random uniform distribution function. The random uniform function spreads the treatment isocenter position uniformly throughout the treatment volume with the location of the center of the radiation isocenters limited to the physical extent of the target.
- The number of generation parameter specifies the maximum number of iterations before the algorithm halts.
- The stall generations is the number of iterations before the algorithm stops if there is no improvement in the objective function value for consecutive generations
- The weighting vector for determining the composite cost function value has a length of eight that contains the importance of the conformity indices and plan quality criteria discussed in section 3.5. The structure of the vector is: 1) Percent

isodose coverage importance, 2) RTOG coverage index importance, 3) RTOG conformity index importance, 4) Paddick conformity index importance, 5) Gradient index importance, 6) Not Used, 7) Not Used & 8) Critical structure tolerance importance.

- Fitness scaling converts the raw fitness scores that are returned by the fitness function to values in a range that is suitable for a selection function to determine the best performing members of the population generation. Fitness scaling of rank and proportional are available for the optimization. Rank fitness scaling results in a larger variance of the population individuals early in the optimization allowing the problem space to be search more thoroughly. If using proportional, the range of the scaled values affects the performance of the genetic algorithm because individuals with the highest scaled values can reproduce more rapidly limiting the ability of the optimization to search other areas of the solution space. Conversely, if the scaled values vary only a little, all individuals have approximately the same chance of reproduction and the search will progress very slowly. Rank fitness scaling removes the effect of the spread of the raw scores and is mainly utilized in this work.
- Selection functions use the scaled fitness values to select the parents of the next generation by assigning a higher probability of selection to individuals with higher scaled values. Stochastic uniform is used in this work in which the function lays out a line where each parent corresponds to a section of the line proportional to its scaled value. The algorithm then divides the line equally by the

- number of individuals to be selected. At each division of the line the algorithm allocates a parent from the section of the line that corresponds to the division.
- The crossover fraction is the portion of the next population generation created by combining the vectors of a pair of parents from the current generation as discussed in section 2.4. Crossover children are produced by vector entries from a pair of individuals in the current generation and combining them to form a child. A uniform crossover function is used to produce children that results in each gene having an equal chance of coming from either parent. As the vectors representing the population individuals contain both binary and continuous variables it was necessary to write a custom scattered crossover function to account for the data structure.
 - The elite fraction is the number of individual in the current population that are passed to the children population unchanged as discussed in section 2.4. Elite children are the individuals in the current generation with the best fitness values.
 - Mutation scale and shrink are parameters that define how individuals are mutated to produce children for subsequent generations. Mutation children are created by applying random changes to a single individual in the current generation to create a child. For the continuous variables, the mutation is created by adding a random number chosen from a Gaussian distribution, to each continuous component. The binary mutation is created by comparing a random generated number to a threshold. If the value exceeds the threshold then the current value of the binary component is changed. The scale and shrink parameters are used to control the amount of mutation used to create the child. The mutation scale controls the

standard deviation of the mutation at the first generation, which is the scale multiplied by the range of the initial population for the parameter. The mutation shrink controls the rate at which the average amount of mutation decreases. The standard deviation decreases linearly so that its final value equals one minus the shrink value times its initial value at the first generation. For example, if the shrink has a value of one, then the amount of mutation decreases to zero at the final generation. As the vectors representing the population individuals contain both binary and continuous variables it was necessary to write a custom mutation function to account for the data structure.

3.7 Gradient Descent Hybrid Optimization Module

As previously discussed, Genetic algorithms are capable (there is no guarantee) of rapidly locating the region where the global minimum exists, however they take a relatively long time to locate the exact local optimum in the region of convergence [50]. A hybrid module is utilized to combat this issue by using a conjugate gradient descent local optimizer to improve the result of the genetic algorithm optimized solution by further refining the values of the continuous variables representing the optimal treatment plan. The hope is that the local search method can locate the global minimum of the region where the genetic algorithm optimization located as optimum.

After the genetic algorithm optimization is completed the best performing individual is extracted from the population. The continuous variables that represent this individual (isocenter position and weight) are then further optimized by the MatLab optimization function FMINUNC, a function from the optimization toolbox. FMINUNC is an

unconstrained nonlinear optimizer that uses conjugate gradient methods to minimize a set of continuous variables.

4 Results

The following chapter contains results from the application of the developed system to several Gamma Knife treatment planning cases. Three Gamma Knife treatment optimization cases; a solitary metastasis, an acoustic schwannoma, and a pituitary adenoma are completely reviewed, illustrating the complete functionality of the developed system. The results and details of each optimization are presented individually followed by a comparison of the system result to that of an expert planner and an optimization model provided with the Gamma Plan treatment planning system. The genetic algorithm optimization was performed on a 3.02 GHz Intel Core 2 Duo Macintosh computer with 4.0 GB of RAM.

4.1 Solitary Metastasis Plan Optimization

The developed system was used to optimize the treatment of a left parietal lobe solitary metastasis. Figure 38 displays the rather large delineated target volume outlined in orange in the axial, coronal, and sagittal planes of the planning MRI. The radiation prescription for the target volume was 18 Gray to the 50% isodose line for a maximum dose of 36 Gray. The solitary metastasis has a volume of 7.27 cubic centimeters with the largest dimension in the X, Y, and Z dimension of 2.7 cm, 2.4 cm, and 2.3 cm. These dimensions approach the treatment target size limits for Gamma Knife radiosurgery of 3.0 cm. The volume is spherical in shape with a mean distance from the center of mass to the boundary points of 1.2 cm and the variance to the distances of 0.0088. The principal components of the volume are 2.2 cm, 2.0 cm, and 1.6 cm. The solitary metastasis is distal to any particular cranial critical structure thus there is no particular tissue with a

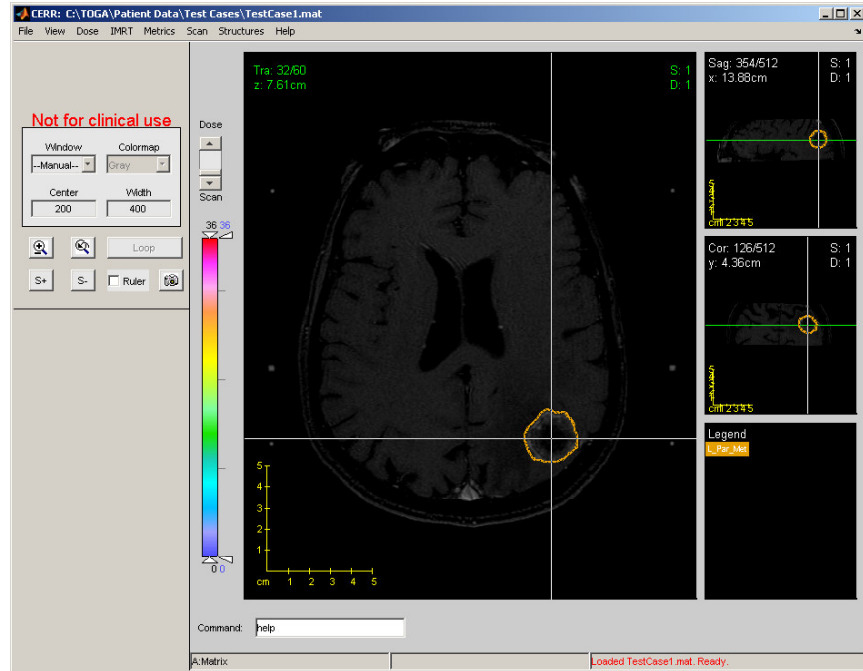


Figure 38 Solitary Metastasis Target Volume

specific dose tolerance. The ideal treatment plan for this target volume will cover the target volume with the prescribed dose of 18 Gray while being as conformal as possible. It is important that the plan be particularly conformal for this treatment volume due to the relatively large size of the diseased areas.

4.1.1 Solitary Metastasis Initial Population Generation

The developed initialization module was utilized to produce a set of treatment plans to serve as the starting point for the genetic algorithm optimization. First a collection of similar historical plans were extracted from the database by comparing the target volume, target principal components, and target irregularity parameter to the database. Ten plans were extracted with similarity values ranging from 4.85 to 4.15. The plans were

transformed to the current target volume by a genetic algorithm that maximized the percent isodose coverage of each historical plan. The total historical plan initialization time was 8.6 minutes. The Paddick conformity indices of the ten historical plans ranged from 0.37 to 0.73 with an average of 0.58, and the percent isodose coverage ranged from 69.5% to 90.9% with an average of 81.1% indicating a high quality set of initial plans.

To supplement the set of historical plans, five geometrically optimized plans were created. The collimator size was limited to the maximum of 18 mm for three of the plans and 14 mm for the other two plans. The coverage parameter was set to 0.7, 0.6, and 0.5 for the 18 mm maximum collimator size plans and 0.5 and 0.45 for the 14 mm maximum collimator size plans. The total time for the geometric optimization was 2.1 minutes. The Paddick conformity indices of the five geometric optimization plans ranged from 0.30 to 0.64 with an average of 0.49, and the percent isodose coverage ranged from 30.1% to 72.4% with an average of 51.5%. The geometric optimization did not produce as good a set of initialization plans as the historical plan set, however the genetic algorithm optimizer may be able to glean some information from the set of plans, particular information regarding the useful collimator sizes, so the geometric optimized plans were included in the initial population.

4.1.2 Solitary Metastasis Genetic Algorithm Optimization

An optimized Gamma Knife treatment plans was produced for the solitary metastasis using the genetic algorithm optimizer with the set of created initial plans as the starting point. The genetic algorithm was allowed to run for 100 generations with a population size of 15 (the number of created initial plans). The elite fraction was set to 2, the

crossover fraction was set to 0.7, the mutation scale and shrink were set to 1, the fitness scaling was set to rank, and the selection function was set to stochastic uniform. The weighting vector for specifying the importance of the various conformity indices was set so that the composite cost value was a weighted average of the percent isodose coverage, the Paddick conformity index, and the Gradient index with weights of 0.7, 1.0, and 0.05 respectively. These weighting values were chosen to match the planning objective of a highly conformal treatment plan with high isodose coverage based on extensive experimentation with the developed code. The genetic algorithm completed the optimization in 9.5 minutes and the final cost value of the optimized treatment plan was 0.158. The progress of the optimization can be seen in figure 39 and the optimized isodose distribution can be seen in figure 40 overlaid on the planning MRI. As evident from figure 40 the optimized plan isodose distribution is conformal to the target volume. Table 7 contains the treatment plan information for the ten isocenter optimized plan. The optimized plan features two 4 mm, six 8 mm, and two 18 mm collimators. The conformity indices for the optimized plan were percent isodose coverage of 90.8%, RTOG coverage index of 0.528, an RTOG conformity index of 1.027, Paddick conformity index of 0.8032, and a Gradient index of 2.78. Each of these indices is acceptable for treatment especially the high Paddick index that indicates that the plan is exceedingly conformal. Ideally the percent isodose coverage and RTOG coverage index value would be slightly higher, preferable greater than 95% and 0.9 respectively. A review of the isodose distribution shows that the genetic algorithm optimization did not completely cover a small superior and inferior portion of the target volume. Perhaps if

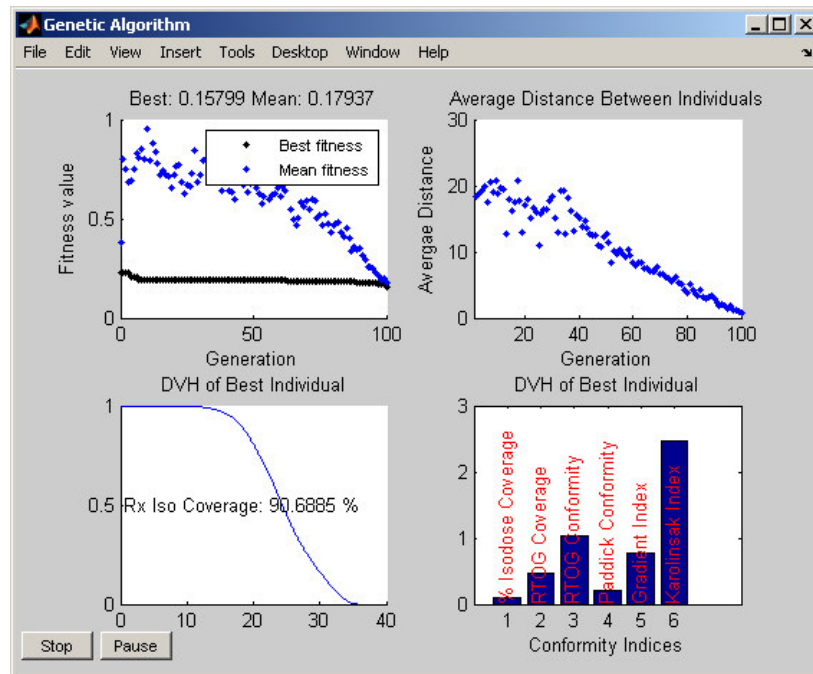


Figure 39 Solitary Metastasis Optimization

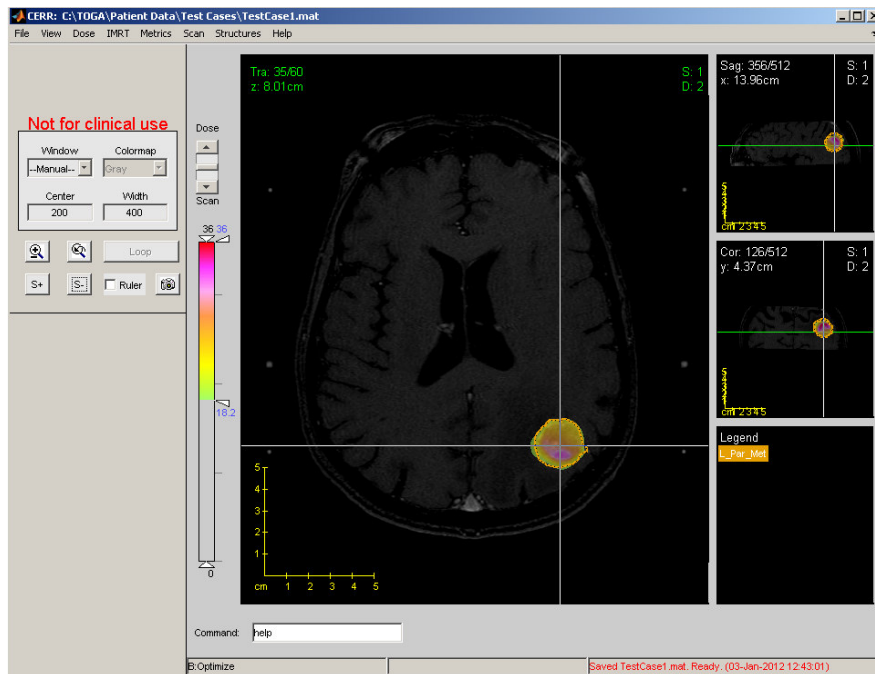


Figure 40 Solitary Metastasis Optimization Isodose Coverage

Table 7 Solitary Metastasis Treatment Plan Parameters

Collimator	X	Y	Z	Weight	γ Angle
4	14.9	4.0	8.0	0.34	90
4	16.2	3.7	8.8	0.46	90
8	13.3	3.8	8.4	0.52	90
8	13.3	4.6	6.5	0.38	90
8	13.8	3.5	7.1	0.77	90
8	14.1	3.7	8.0	0.59	90
8	14.6	4.4	7.1	0.53	90
8	14.2	4.1	6.6	0.75	90
18	14.0	4.7	7.7	1.57	90
18	12.8	4.3	7.1	0.75	90

the algorithm had been allowed to run for more generations it would have positioned an isocenter in these areas of the target volume, however in the interest of time the algorithm was only allowed to run for 100 generations. To further improve the genetic algorithm optimized result, two 8 mm isocenter was manually added to the treatment plan at the under treated areas and given weights of 0.6. This manual revision improved the percent isodose coverage to 95.22% and the Paddick conformity index of 0.83 producing an excellent treatment plan that is highly conformal and adequately covers the target volume. This manual revision took less than one minute to perform and suggests there might be some utility in periodically inspecting the result during the optimization, make some manual improvement, and then allow the optimizer to continue. This plan revision could also be coded to keep with the objective of producing a completely automated optimization tool as part of the future work of the project.

4.1.3 Solitary Metastasis Optimization Compared to Expert Planner

To compare the genetic algorithm optimized plan to a typical manually produced plan, an expert planner designed a treatment plan for the solitary metastasis while being timed.

The expert planner, also the author of this dissertation, has seven years experience with Gamma Knife treatment planning and has produced 652 Gamma Knife treatment plans when this dissertation was written, 519 of which were for the Gamma Knife Model C.

The expert planner developed an eleven treatment isocenter plan consisting of one 18 mm and ten 8 mm collimator isocenters in 23.6 minutes. Figure 41 shows a comparison window of the genetic algorithm optimized plan (shown in blue) and the expert planners plan (shown in green). The expert's plan has a better percent isodose coverage (1.5% higher) and Gradient index (0.28 lower) while the genetic algorithm

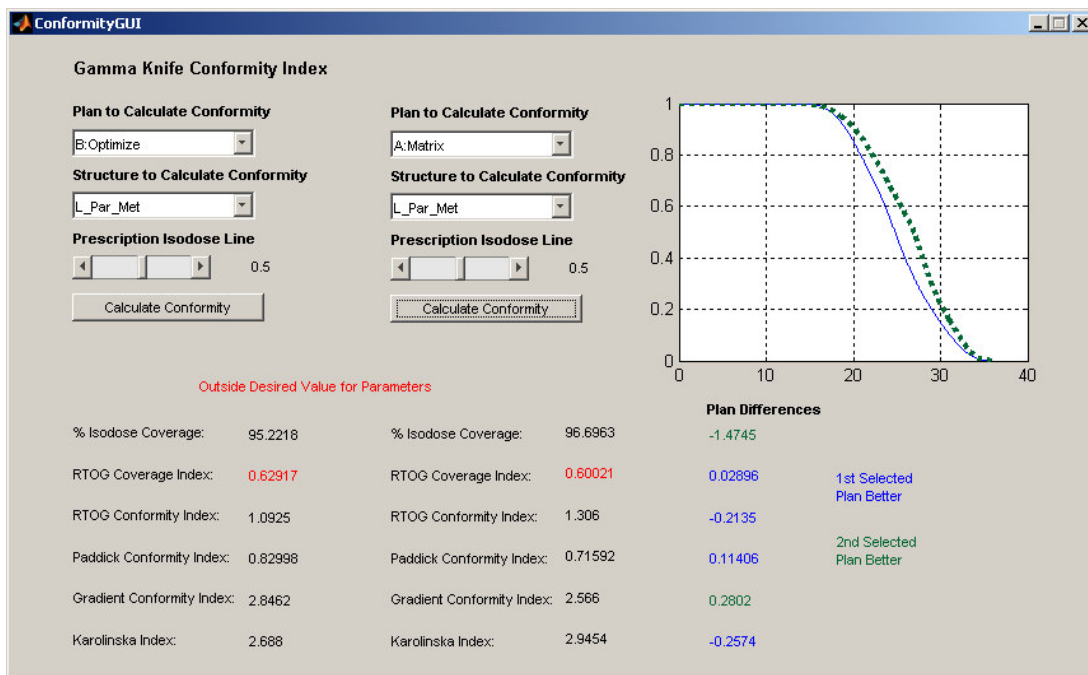


Figure 41 Solitary Metastasis Optimization Plan versus Expert Planner

optimized plan had a better Paddick Conformity index (0.11 higher), RTOG Coverage index (0.02) and RTOG Conformity Index (0.21 lower). The two plans are very similar in quality with the expert planner taking 3.4 minutes longer to produce the plan.

4.1.4 Solitary Metastasis Optimization Compared to Gamma Plan

To compare the genetic algorithm optimized plan to another Gamma Knife optimizing system, a treatment plan was produced by the Gamma Plan optimizer included with the treatment planning software. The Gamma Plan optimizer uses a skeletonization technique as describe in section 2.2.5 of the literature survey followed by a gradient optimizer. The system optimizes extremely quickly with typical optimization times of about a minute. The skeletonization optimization requires the user to select a shot size and a packing parameter that describes how closely to place the treatment isocenter to one another along the skeleton of the structure. The user runs the optimizer for each collimator size that is to be included in the plan generally starting with the largest collimator. After a plan is produced with the skeletonization optimizer the user can run a gradient optimizer to improve the plan quality. The Gamma Plan optimizer was used to produce a treatment plan consisting of one 18 mm and ten 8 mm collimator isocenters for the solitary metastasis. The skeletonization and gradient descent optimizer ran for about one minute to produce an optimal result. Figure 42 shows a comparison window of the genetic algorithm optimized plan (shown in blue) and the Gamma Plan optimizer plan (shown in green). The genetic algorithm optimized plan had as a better percent isodose coverage (6.9% higher), Paddick Conformity Index (0.08 higher), and better RTOG Coverage Index (0.07 higher) while the Gamma Plan optimized plan had a better Gradient Index (0.27 lower) and RTOG Conformity index (0.05). The genetic algorithm

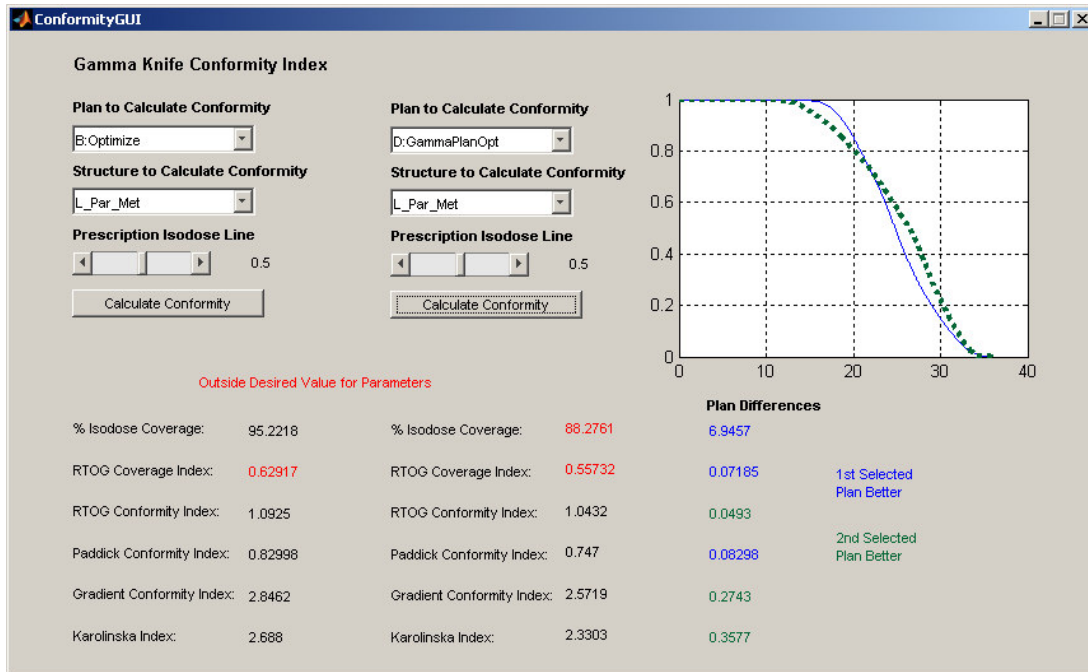


Figure 42 Solitary Metastasis Optimization Plan versus Gamma Plan Optimizer

optimized plan is a better treatment plan; however the Gamma Plan optimized plan was produced much more quickly.

4.2 Acoustic Schwannoma Plan Optimization

The developed system was also used to optimize the treatment of a left acoustic schwannoma tumor. Figure 43 displays the delineated target volume outlined in orange in the axial, coronal, and sagittal planes of the planning MRI. The radiation prescription for the target volume was 12.5 Gray to the 50% isodose line for a maximum dose of 25 Gray. The acoustic tumor has a volume of 2.84 cubic centimeters with the largest dimension in the X, Y, and Z dimension of 1.9 cm, 2.2 cm, and 1.5 cm. The volume is spherical in shape exclusive of a tail that extends laterally that is characteristic of acoustic tumors. The mean distance from the center of mass to the boundary points of

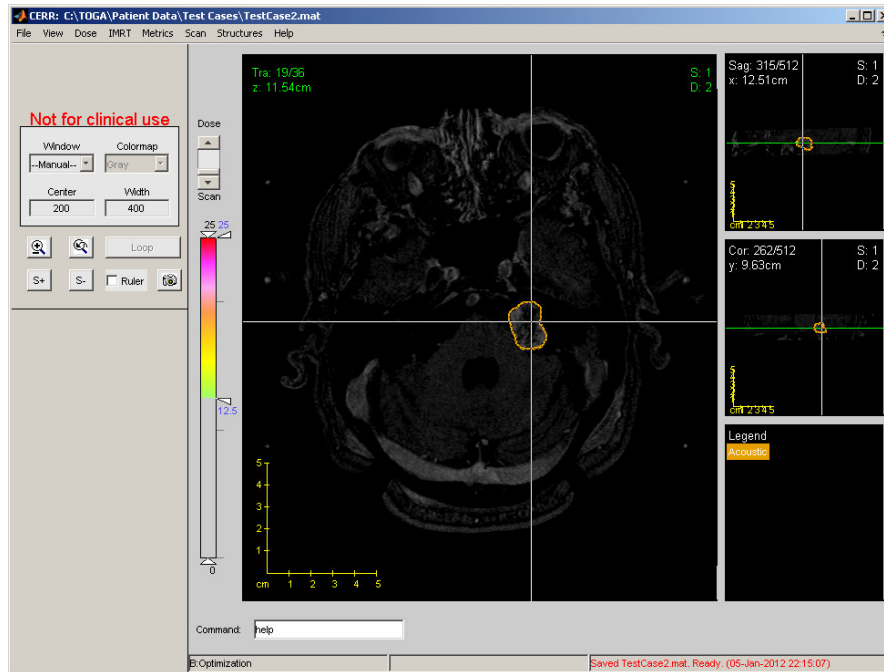


Figure 43 Acoustic Schwannoma Target Volume

0.92 cm and the variance of the distances is 0.0239. The principal components of the volume are 1.9 cm, 1.4 cm, and 1.1 cm. The tumor volume is adjacent medially to the brainstem and laterally to the cochlea thus the radiation dose to these structures should be evaluated and limited to 12 Gray maximum for the brainstem and ideally 4 Gray mean dose to the cochlea if possible depending on patient anatomy. The ideal treatment plan will cover the target volume with the prescribed dose of 12.5 Gray while being as conformal particularly limiting the dose of the brainstem and the cochlea.

4.2.1 Acoustic Schwannoma Initial Population Generation

The developed initialization module was utilized to produce a set of treatment plans to serve as the starting point for the genetic algorithm optimization. First a collection of similar historical plans were extracted from the database by comparing the target volume,

target principal components, and target irregularity parameter to the database. Fifteen plans were extracted with similarity values ranging from 4.87 to 4.1. The plans were transformed to the current target volume by a genetic algorithm that maximized the percent isodose coverage of each historical plan. The total historical plan initialization plan time was 3.5 minutes. The Paddick conformity indices of the ten historical plans ranged from 0.26 to 0.56 with an average of 0.42, and the percent isodose coverage ranged from 50.3% to 92.4% with an average of 70.1%. The collection of historical plans did not have extremely high conformity indices but at least one of the plans has good target coverage and the set should serve as a good starting point for the optimization.

To supplement the set of historical plans five geometrically optimized plans were created. The collimator size was limited to the maximum of 14 mm for two of the plans and 8 mm for the other three plans. The coverage parameter was set to 0.65 and 0.6 for the 14 mm maximum collimator size plans and 0.55, 0.5, and 0.45 for the 8 mm maximum collimator size plans. The total time for the geometric optimization was 1.7 minutes. The Paddick conformity indices of the five geometric optimization plans ranged from 0.40 to 0.59 with an average of 0.53, and the percent isodose coverage ranged from 59.5% to 70.1% with an average of 62.9%. The geometric optimization did not produce a set of initial plans with particularly high target coverage however the plans are more conformal than the historical database so the genetic algorithm optimizer should be able to extract some useful information from the plan set.

4.2.2 Acoustic Schwannoma Genetic Algorithm Optimization

An optimized Gamma Knife treatment plan was produced for the acoustic schwannoma using the genetic algorithm optimizer with the set of created initial plans as the starting point. The genetic algorithm was allowed to run for 200 generations with a population size of 20 (the number of created initial plans). The elite fraction was set to 2, the crossover fraction was set to 0.8, the mutation scale and shrink were set to 1, the fitness scaling was set to rank, and the selection function was set to stochastic uniform. The weighting vector for specifying the importance of the various conformity indices was set so that the composite cost value was a weighted average of the percent isodose coverage, the Paddick conformity index, and the Gradient index with weights of 0.8, 1.5, and 0.05 respectively. These weighting values were chosen to match the planning objective of a highly conformal treatment plan with high isodose coverage based on extensive experimentation with the developed code. The genetic algorithm optimization ended after the stall generation was reached after 125 generations. The optimization took 11.5 minutes and the final cost value of the optimized treatment plan was 0.216. The progress of the optimization can be seen in figure 44 and the optimized isodose distribution can be seen in figure 45 overlaid on the planning MRI. Table 8 contains the treatment plan information for the ten isocenter optimized plan. The optimized plan features five 4 mm, three 8 mm, and two 14 mm collimators. The conformity indices for the optimized plan were percent isodose coverage of 95.3%, RTOG coverage index of 0.73, an RTOG conformity index of 1.17, Paddick conformity index of 0.77, and a Gradient index of 2.6. Each of these indices is acceptable for treatment especially the high percent isodose coverage and Paddick Conformity index

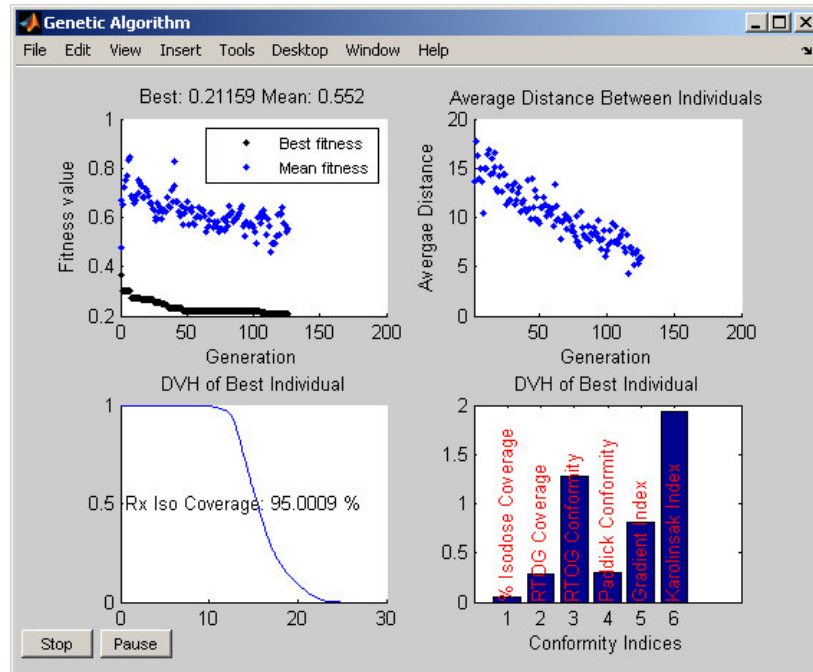


Figure 44 Acoustic Schwannoma Optimization

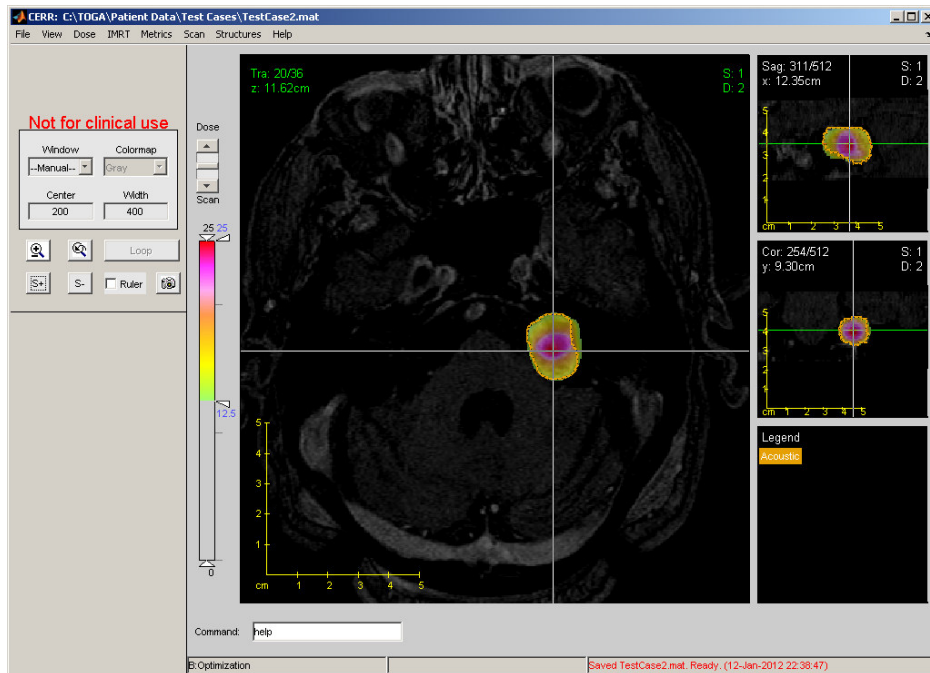


Figure 45 Acoustic Schwannoma Optimization Isodose Coverage

Table 8 Acoustic Schwannoma Treatment Plan Parameters

Collimator	X	Y	Z	Weight	γ Angle
4	11.9	8.8	12.2	1.25	90
4	14.2	10.4	9.8	1.45	90
4	13.1	6.8	12.7	1.17	90
4	11.3	8.3	12.4	0.55	90
4	12.3	9.2	11.7	0.59	90
8	12.4	8.9	12.3	1.14	90
8	13.0	7.7	11.8	0.21	90
8	13.1	10.2	12.0	1.50	90
14	12.4	9.1	11.6	2.97	90
14	12.3	9.8	11.6	2.19	90

that indicates that the plan is exceedingly conformal. A review of the isodose distribution show the genetic algorithm was able to produce a highly conform plan that adequately covered the target volume. Also the radiation dose to the brainstem did not exceed 12 Gray and the dose to the cochlea did not exceed 4 Gray.

4.2.3 Acoustic Schwannoma Optimization Compared to Expert Planner

To compare the genetic algorithm optimized plan to a typical manually produced plan, an expert planner designed a treatment plan for the acoustic schwannoma while being timed. The expert planner developed a six isocenter treatment plan consisting of one 14 mm and five 8 mm collimator isocenters in 12.7 minutes. Figure 46 shows a comparison window of the genetic algorithm optimized plan (shown in blue) and the expert planners plan (shown in green). The expert's plan has a better percent isodose coverage (2.4% higher) while the genetic algorithm optimized plan had a better Paddick

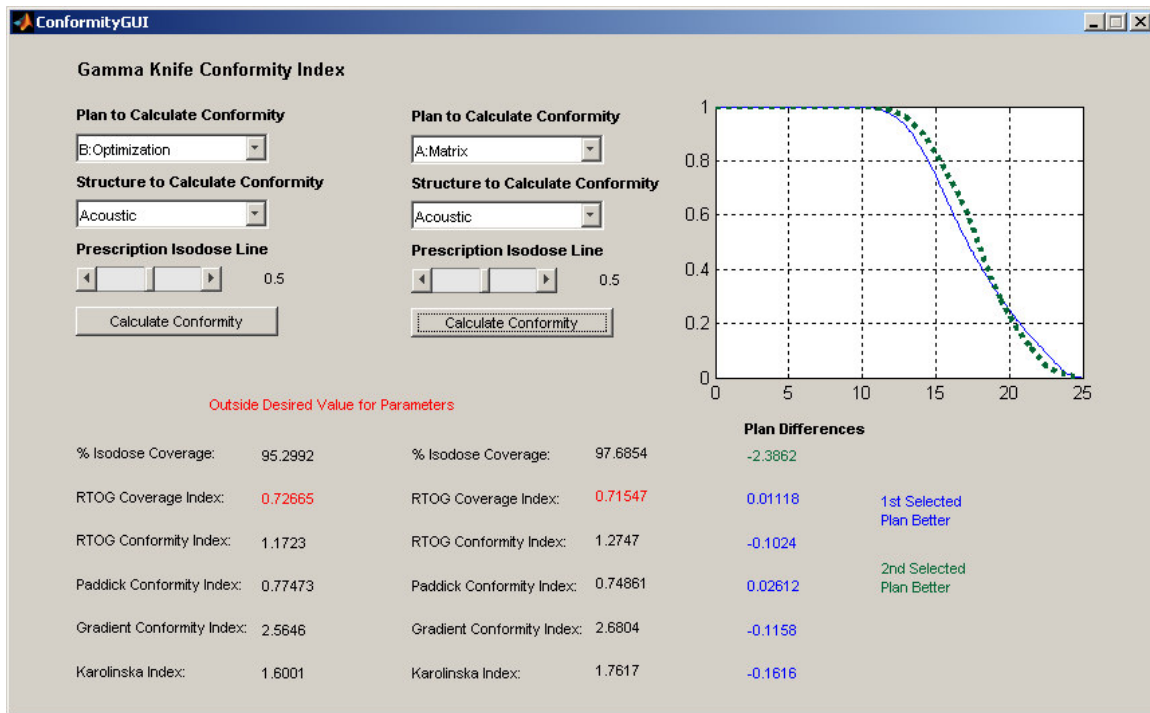


Figure 46 Acoustic Schwannoma Optimization Plan vs. Expert Planner

Conformity index (0.03 higher), RTOG Coverage index (0.01 higher), RTOG Conformity Index (0.10 lower), and a lower Gradient index (0.1 lower). The genetic algorithm plan is higher in quality however the experts plan was produced four minutes quicker and contains fewer isocenters.

4.2.4 Acoustic Schwannoma Optimization Compared to Gamma Plan

To compare the genetic algorithm optimized plan to another Gamma Knife optimizing system a treatment plan was produced by the Gamma Plan optimizer included with the treatment planning software. The Gamma Plan optimizer was used to produce a treatment plan consisting of five 8 mm and four 4 mm collimator isocenters for the acoustic schwannoma. The skeletonization and gradient descent optimizer ran for about

one minute to produce an optimal result. Figure 47 shows a comparison window of the genetic algorithm optimized plan (shown in blue) and the Gamma Plan optimizer plan (shown in green). The genetic algorithm optimized plan had as a better percent isodose coverage (7.2% higher), Paddick Conformity Index (0.04 higher), and better RTOG Coverage Index (0.24 higher) while the Gamma Plan optimized plan had a better RTOG Conformity index (0.11). The Gradient index was equivalent between the two systems. The genetic algorithm optimized plan is a better treatment plan; however the Gamma Plan optimized plan was produced much more quickly.

4.3 Pituitary Adenoma Plan Optimization

The developed system was used to optimize the treatment of a pituitary tumor. Figure 48 displays the delineated target volume outlined in orange in the axial, coronal,

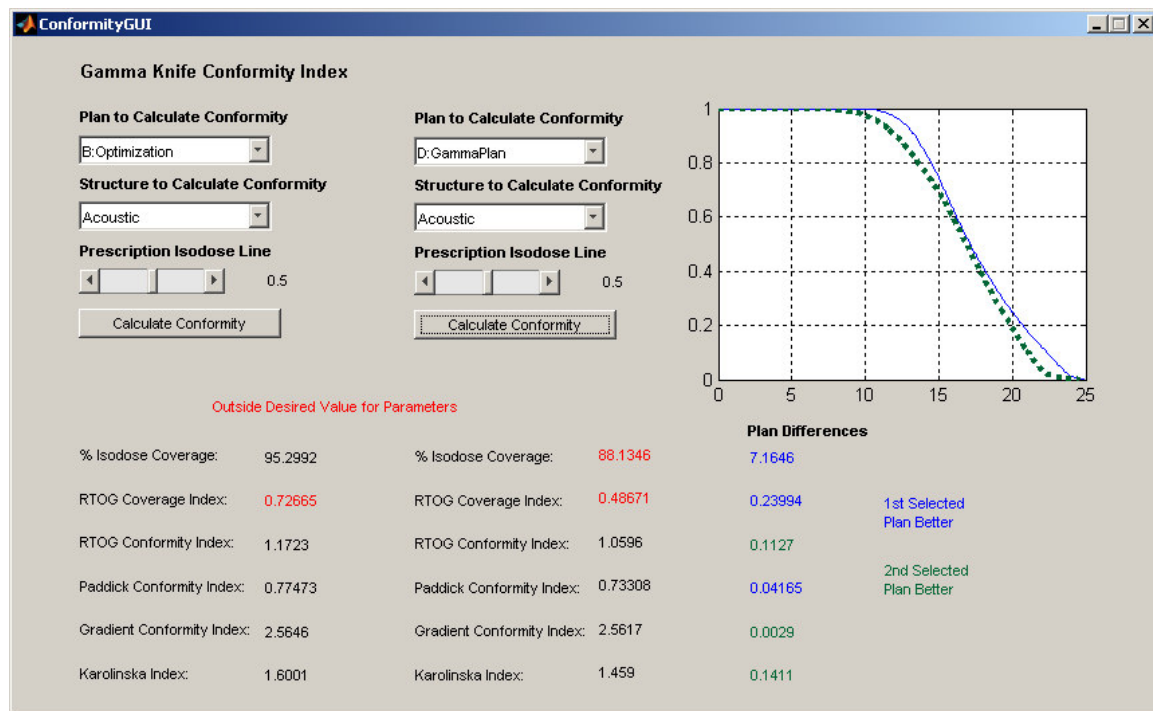


Figure 47 Acoustic Tumor Optimized Plan vs. Gamma Plan Optimizer

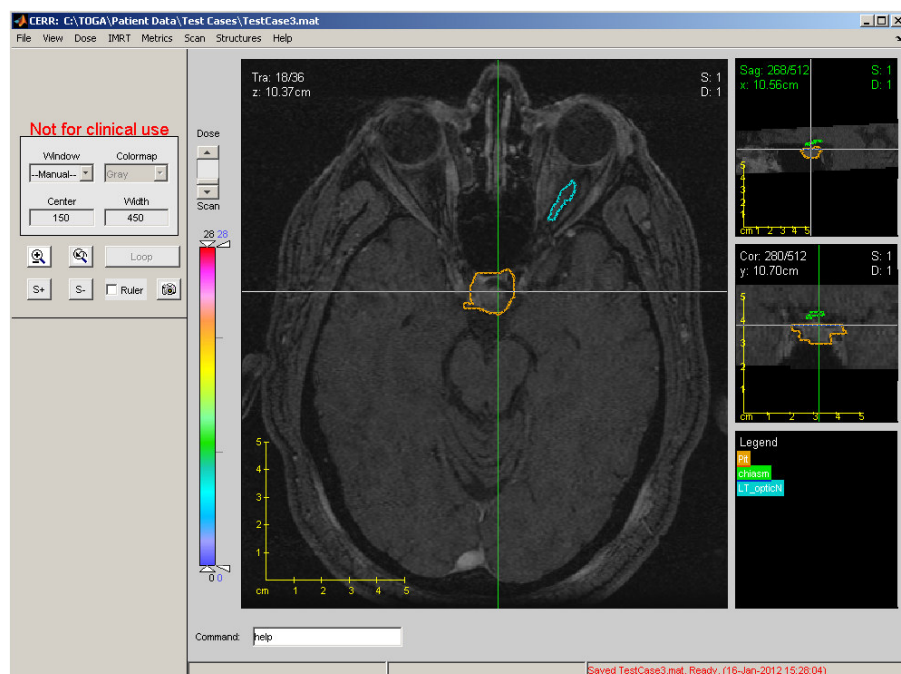


Figure 48 Pituitary Adenoma Target Volume

and sagittal planes of the planning MRI. The radiation prescription for the target volume was 14 Gray to the 50% isodose line for a maximum dose of 28 Gray. The pituitary tumor has a volume of 1.62 cubic centimeters with the largest dimension in the X, Y, and Z dimension of 2.4 cm, 1.8 cm, and 0.8 cm. The volume is spherical in shape with a mean distance from the center of mass to the boundary points of 0.8 cm and the variance of the distances is 0.04. The principal components of the volume are 1.9 cm, 1.5 cm, and 0.6 cm. The tumor volume is located directly inferior to the optic chiasm and medially and inferior to both the left and right optic nerves. The maximum radiation dose to the each of these optic structures should be limited to 8 Gray to avoid injury. The ideal treatment plan will cover the target volume with the prescribed dose of 14 Gray while being as conformal particularly limiting the dose of the optic structures.

4.3.1 Pituitary Adenoma Initial Population Generation

The developed initialization module was utilized to produce a set of treatment plans to serve as the starting point for the genetic algorithm optimization. First a collection of similar historical plans were extracted from the database by comparing the target volume, target principal components, and target irregularity parameter to the database. Nine plans were extracted with similarity values ranging from 4.2 to 3.6. The plans were transformed to the current target volume by a genetic algorithm that maximized the percent isodose coverage of each historical plan. The total historical plan initialization plan time was 2.6 minutes. The Paddick conformity indices of the ten historical plans ranged from 0.19 to 0.49 with an average of 0.33, and the percent isodose coverage ranged from 54.7% to 93.8% with an average of 78.9%. The collection of historical plans did not have extremely high conformity indices but did have high isodose target coverage and should serve as a good starting point for the optimization.

To supplement the set of historical plans five geometrically optimized plans were created. The collimator size was limited to the maximum of 8 mm for four of the plans and 4 mm for the other plan. The coverage parameter was set to 0.65, 0.6, 0.55 and 0.5 for the 8 mm maximum collimator size plans and 0.45 for the 4 mm maximum collimator size plans. The total time for the geometric optimization was 2.3 minutes. The Paddick conformity indices of the five geometric optimization plans ranged from 0.28 to 0.44 with an average of 0.37, and the percent isodose coverage ranged from 35.6% to 69.4% with an average of 55.7%. The geometric optimization did not produce a set of initial plans with particularly high target coverage however the plans are more conformal than

the historical database so the genetic algorithm optimizer should be able to extract some useful information from the plan set.

4.3.2 Pituitary Adenoma Genetic Algorithm Optimization

An optimized Gamma Knife treatment plans was produce for the pituitary adenoma using the genetic algorithm optimizer with the set of created initial plans as the starting point. The genetic algorithm was allowed to run for 100 generations with a population size of 14 (the number of created initial plans). The elite fraction was set to 2, the crossover fraction was set to 0.8, the mutation scale and shrink were set to 1, the fitness scaling was set to rank, and the selection function was set to stochastic uniform. The weighting vector for specifying the importance of the various conformity indices was set so that the composite cost value was a weighted average of the percent isodose coverage, the Paddick conformity index, the Gradient index and the critical structure tolerance set to a maximum dose of 8 Gray. The weight for each of these parameters was 1, 1, 0.05 and 0.07 respectively. These weighting values were chosen to match the planning objective of a highly conformal treatment plan with high isodose coverage based on extensive experimentation with the developed code. The optimization took 9.6 minutes and the final cost value of the optimized treatment plan was 0.186. The progress of the optimization can be seen in figure 49 and the optimized isodose distribution can be seen in figure 50 overlaid on the planning MRI. Table 9 contains the treatment plan information for the nine isocenter optimized plan. The optimized plan features three 4 mm, and six 8 mm collimators. The conformity indices for the optimized plan were percent isodose coverage of 92.8%, RTOG coverage index of 0.51, an RTOG conformity index of 1.27, Paddick conformity index of 0.68, and a Gradient index of 2.7.

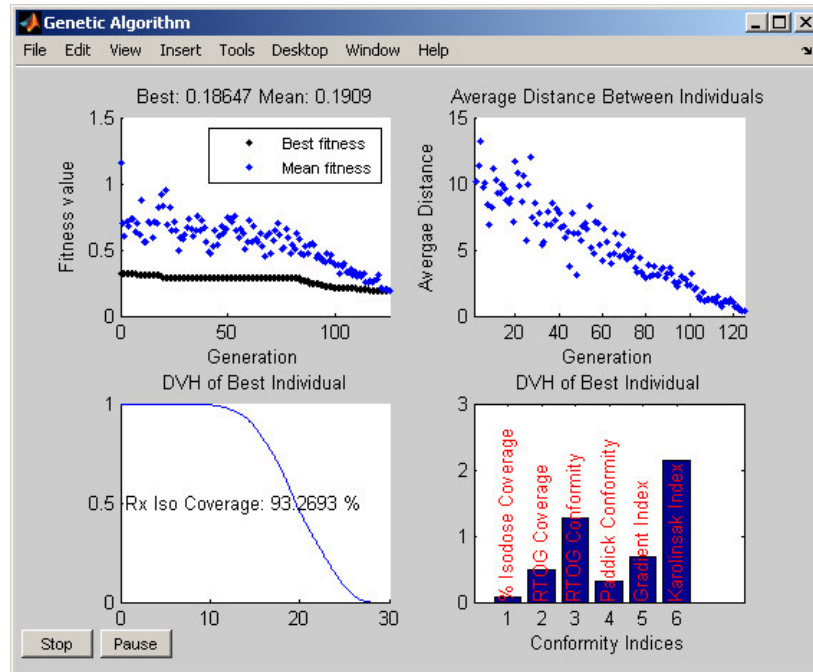


Figure 49 Pituitary Adenoma Optimization

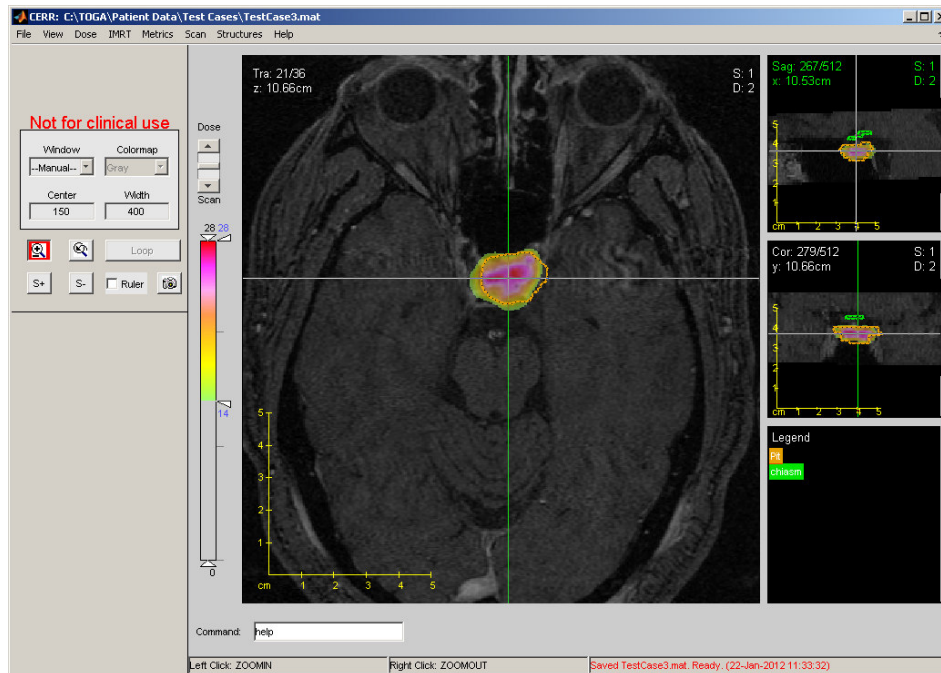


Figure 50 Pituitary Adenoma Optimization Isodose Coverage

Table 9 Pituitary Adenoma Treatment Plan Parameters

Collimator	X	Y	Z	Weight	γ Angle
4	11.2	11.4	10.5	0.86	72
4	11.2	9.9	10.6	0.11	72
4	11.6	11.0	10.7	0.20	72
8	10.1	11.0	10.6	1.00	72
8	9.5	10.4	10.6	0.65	72
8	11.0	11.0	10.8	0.93	72
8	10.1	10.4	10.7	0.70	72
8	11.0	10.6	10.8	0.72	72
8	10.7	9.9	10.6	0.70	72

The maximum dose to the optic chiasm, right optic nerve, and left optic nerve was 7.5, 3.2, and 4.7 Gray respectively. Figure 51 displays a dose volume histogram plot for each of the critical structures and the pituitary tumor target volume. Each of the conformity indices is acceptable for treatment especially considering the low dose to the optic structures in close proximity to the target volume. A review of the isodose distribution show the genetic algorithm was able to produce a highly conform plan that adequately covered the target volume.

4.3.3 Pituitary Adenoma Optimization Compared to Expert Planner

To compare the genetic algorithm optimized plan to a typical manually produced plan, an expert planner designed a treatment plan for the pituitary adenoma while being timed. The expert planner developed an eleven isocenter treatment plan consisting of three 8 mm and eight 4 mm collimator isocenters in 25.3 minutes. Figure 52 shows a comparison window of the genetic algorithm optimized plan (shown in blue) and the

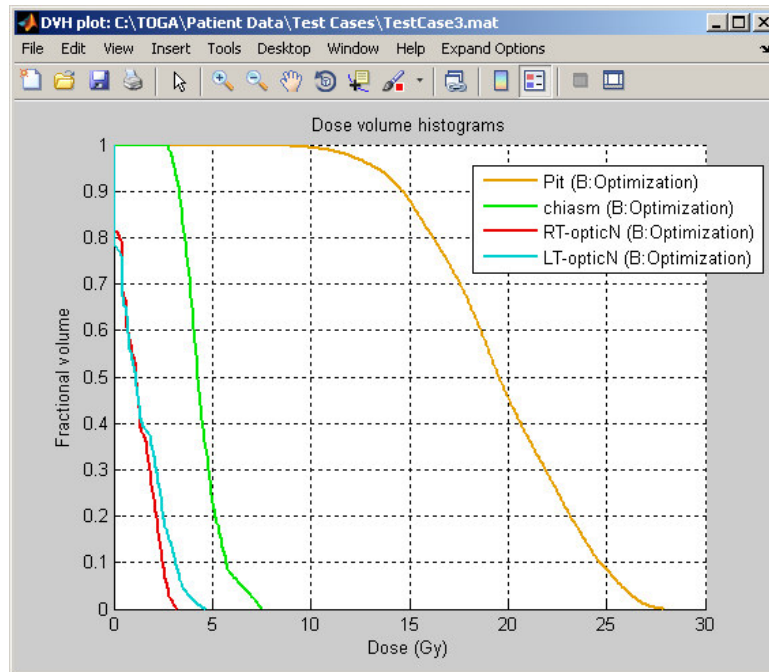


Figure 51 Dose Volume Histogram of Pituitary Adenoma Treatment Plan

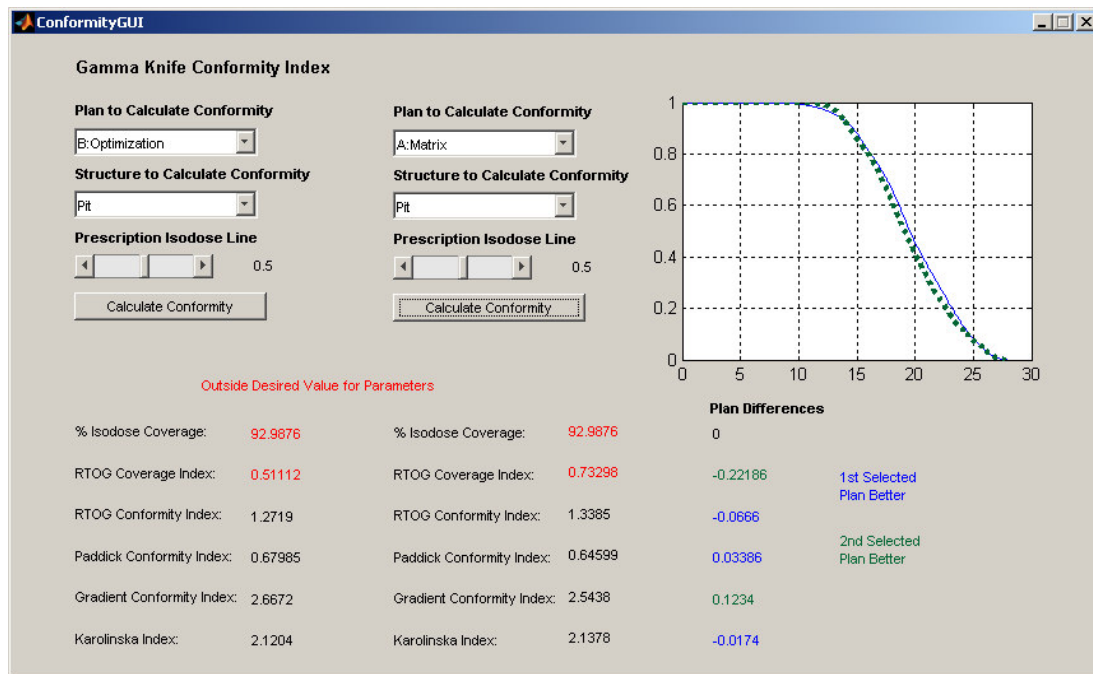


Figure 52 Pituitary Adenoma Optimization Plan vs. Expert Planner

expert planners plan (shown in green). The expert's plan has a better RTOG Coverage Index (0.22 higher) and a lower Gradient index (0.12 lower) while the genetic algorithm optimized plan had a better Paddick Conformity Index (0.03 higher) and a higher RTOG Conformity Index (0.06 lower). The two plans had equivalent percent isodose coverage indices of 92.8%. Both plan limited the maximum radiation dose to the optic structures to less than 8 Gray. The two treatment plans are essentially equivalent in quality with the genetic algorithm plan being produced 10 minutes quicker and containing two fewer isocenter.

4.3.4 Pituitary Adenoma Optimization Compared to Gamma Plan

To compare the genetic algorithm optimized plan to another Gamma Knife optimizing system a treatment plan was produced by the Gamma Plan optimizer included with the treatment planning software. The Gamma Plan optimizer was used to produce a treatment plan consisting of three 8 mm and three 4 mm collimator isocenters for the pituitary adenoma. The skeletonization and gradient descent optimizer ran for about one minute to produce an optimal result. Figure 53 shows a comparison window of the genetic algorithm optimized plan (shown in blue) and the Gamma Plan optimizer plan (shown in green). The genetic algorithm optimized plan had as a better percent isodose coverage (12.3% higher), Paddick Conformity Index (0.03 higher), and better RTOG Coverage Index (0.10 higher) while the Gamma Plan optimized plan had a better RTOG Conformity index (0.27 lower) and a lower Gradient index (0.15 lower). The genetic algorithm optimized plan is a better treatment plan, however the Gamma Plan optimized plan was produced much more quickly.

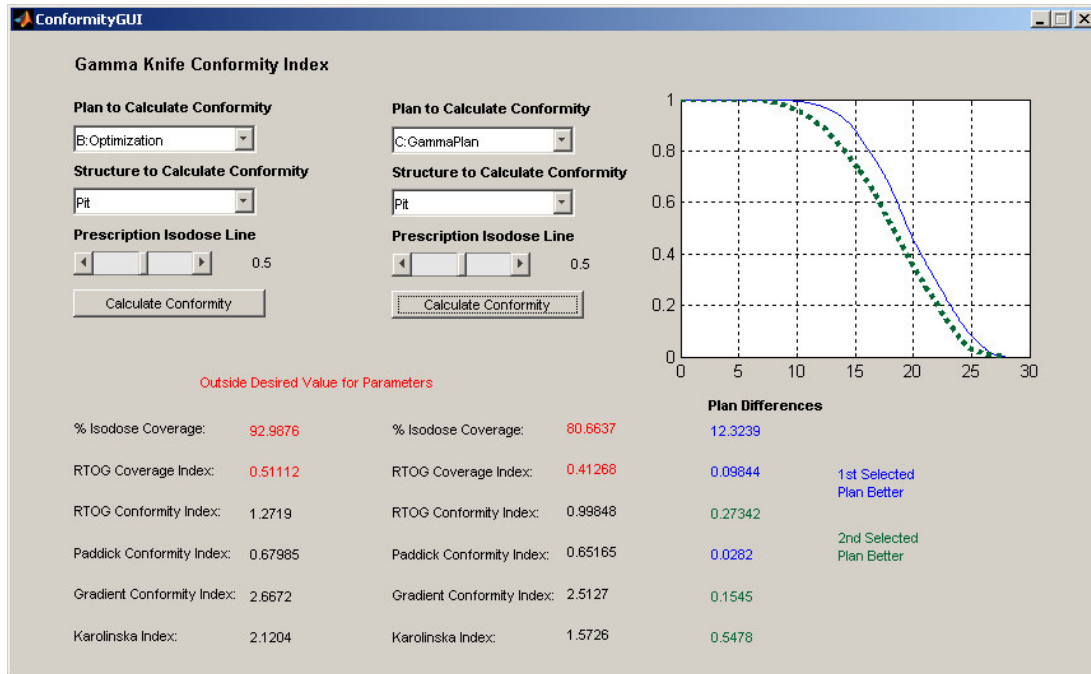


Figure 53 Pituitary Adenoma Optimized Plan vs. Gamma Plan Optimizer

4.4 Summary of Optimization Results

The developed system was successful in optimizing the Gamma Knife treatment plans for the three presented cases. The system essentially performed as well as the expert planner and performed much better than the optimizer included with the Gamma Plan system; however the Gamma Plan optimizer produced a result much more quickly. The system produced an average percent isodose coverage for the three plans of 94.5% and the average Paddick Conformity index was 0.76 in an average time of minutes 17.2 minutes. Table 10 presents a summary of the optimization for each test case.

Table 10 Test Case Summary

		Test Case 1	Test Case 2	Test Case 3
	Tumor Type	Metastasis	Acoustic Schwannoma	Pituitary Adenoma
	Target Volume (cc)	7.27	2.84	1.62
	Target Principal Components (cm)	2.2x2.0x1.6	1.9x1.4x1.1	1.9x1.5x0.6
	Radiation Prescription	18 Gy @ 50%	12.5 Gy @ 50%	14 Gy @ 50%
Expert Planner	Total Optimization Time	23.6	12.7	25.3
	Paddick Conformity Index	0.72	0.75	0.65
	% Isodose Coverage	96.7%	97.7%	92.8%
Gamma Plan Optimizer	Total Optimization Time	1	1	1
	Paddick Conformity Index	0.75	0.73	0.65
	% Isodose Coverage	88.3%	88.1%	80.7%
TOGA	Total Optimization Time	20.2	16.7	14.5
	Paddick Conformity Index	0.83	0.77	0.68
	% Isodose Coverage	95.2%	95.3%	92.8%

5 Conclusions and Suggestions for Future Work

An automated system was developed for performing Gamma Knife treatment planning for stereotactic radiosurgery. The need for such a system was discussed in regards to improving planning times and the likelihood of developing an optimal plan for the designed treatment volume. A complete literature review was presented that outlined other methods for optimizing Gamma Knife treatment planning. The developed work differs in that it is the first that utilizes a collection of historical treatment plans, determined to be similar to the target volume, that are optimized via a genetic algorithm to arrive at an optimal plan for the target volume. Additionally the developed system is almost fully automated exclusive of a weighting vector that specifies the importance of several conformity indices and a few initialization selections. Testing of the developed system demonstrated that it is an effective and efficient planning tool that is capable of optimizing treatment plans quickly to the same degree as an expert planner. The goals of this research have been accomplished through the five original contributions.

5.1 Summary of original contributions

There are five areas of original contribution that were discussed. Each of these original contributions were covered with a thorough literature review and an in depth study of each research topic.

The first original contribution was the development of an initialization module that generates several new treatment plans that are appropriate to treat a physician designated target volume. This was accomplished by utilizing a kernel regression non-parametric

modeling technique to compare a set of extracted parameters that characterized the target volume to a database of historical target volumes and treatment plans. The comparison generates a set of treatment plans that are appropriate for the current treatment. The set of plans is finally tailored to match the target volume by a rigid transformation where the parameters of the transformation were optimized by a genetic algorithm. The work was unique in that it utilized principal component analysis to characterize the target volume in addition to the typical parameters used to define a volume should as diameter and volume. This alleviated the issue of relative geometric orientation of the target volume to the set of database volumes that exist in a past work that featured a data mining module. The developed module was effective at rapidly producing a collection of quality treatment plans to serve as the starting point of a genetic algorithm optimization. Average extraction and transformation times for the cases presented were 6.9 minutes with Paddick Conformity indices of 0.44.

In addition to the historical plan initialization module, a geometric module was developed for providing additional initialization plans. This module was mainly developed to provide the user with some control of the size of the collimator included in the genetic algorithm initial population and to provide a quality starting plan if none of the target volumes in the historical database were especially similar to the current target volume. The historical plan portion of the initialization module selects treatment plans based exclusively on the parameters of the target volume. If desired, the geometric module allows the user to specify collimator size so that inclusion of a collimator can be guaranteed in the genetic algorithm initial population. The developed geometric

initialization module was effective at rapidly producing a collection of quality treatment plans to serve as the starting point of a genetic algorithm optimization. Average optimization times for the cases presented were 2.1 minutes with Paddick conformity indices of 0.46. While these Paddick conformity indices are not suitable for treatment, the treatment plans serve as useful starting points for the genetic algorithm optimization.

The second original contribution was the development of a genetic algorithm optimization module to optimize the seeded treatment plans to produce a single treatment plan that conforms to the designated treatment volume. A genetic algorithm had not been utilized in the optimization of Gamma Knife treatment planning and this work demonstrates that they are capable of optimizing a collection of treatment plans to produce an optimized treatment plan. Average optimization times for the cases presented were 10.2 minutes with Paddick conformity indices of 0.76.

Genetic algorithm optimization of Gamma Knife treatment planning requires computation of many individual treatment plans. As one of the objectives of this work was to perform the optimization as quickly, it was necessary to create an accurate and quick dose calculation algorithm. The third original contribution of the work was the development of a rapid radiation dose distribution for the genetic algorithm optimization computation. The radiation dose transport calculation was performed by a weighted summation of three dimensional matrices representing the dose distribution from individual treatment isocenters. As discussed in the methods chapter, this technique resulted in an accurate and rapid calculation of the radiation dose distribution from

multiple treatment isocenters. An additional use for this methodology would be the independent verification of a Gamma Knife treatment plan.

The fourth original contribution of the work was the development of a gradient descent optimization module to refine the result of the genetic algorithm optimization to ensure that the system as a whole produces the optimal treatment plan. This methodology had been used alone by several authors in the optimization of Gamma Knife treatment planning. The technique was not found to be useful in this work as no improvement in any conformity index or cost function objection value was found while utilizing this technique.

The fifth and final original contribution of the work was the integration of all developed modules into a single computer program that can be quickly and efficiently used to optimize a Gamma Knife treatment plan. As previously discussed it is desirable to develop the Gamma Knife treatment plan as quickly as possible as the patient is waiting for treatment with the stereotactic head frame attached. Additionally the treating physicians and nurses are waiting for treatment so there is labor cost incentive to produce a quality plan quickly. All developed MatLab functions can be easily called from a simple GUI. For the planning cases presented in the results the average total optimization time, including the time to interact with the program and run the functions was 25 minutes. This result was equivalent to that of an expert planner. While no improvement was made in the total planning time, the system was capable to performing as well in the same time frame as an expert planning with over seven years of experience

with Gamma Knife treatment planning. A system capable of this would be extremely useful for a Gamma Knife center that was just opening or one where the planners lacked extensive experience.

5.2 Future Work

The developed work can be effectively used to optimize Gamma Knife treatment planning. Future work for the developed system includes further testing of the system with varying initialization parameter setting. This includes testing of the Genetic Algorithm with varying population size, cross over & elite fraction, mutation scale and shrink, and methods of scaling the population individuals. Also additional testing of various conformity index weighting vectors could be useful to determine an optimal vector. Testing to date found that the user had to tailor the conformity index weighting vector to the specific planning objective of the patient but continued testing could produce a set of useful vectors for various treatment aims. Thus, much could be learned from continued testing and experience using the developed code.

The result for the solitary metastasis indicated it might be beneficial to periodically evaluate the optimization result to determine if a simple manual addition of a treatment isocenter to an under covered area could quickly and dramatically improve the optimization result. Future work should include the development of an optimization evaluation module that stops the optimization at certain intervals, searches the target volume for undertreated areas, adds an isocenter at these areas, and then restarts the optimization with these modified plans.

The work presented here was developed for the Gamma Knife model C. As previously discussed this is the forth generation of the Gamma Knife and a fifth generation known as the Gamma Knife Perfexion was introduced in 2007. The Perfexion was mechanically a major redesign from all previous versions of the Gamma Knife. For the Perfexion the collimators that shape the radiation beam are internal and mechanically exchanged rather than external and manually exchanged for the Gamma Knife C. This change resulted in the Perfexion having three collimator sizes (4, 8, & 16) opposed to the four that the Gamma Knife C has. Additionally, the Perfexion has the ability to make hybrid collimators as the device features 8 collimator sectors rather than one rigid collimator helmet. The Perfexion presents an exciting new application for the optimization system developed in this work. Future work will include the application of the developed work to the optimization of Gamma Knife Perfexion planning optimization. This will be accomplished by first modifying the rapid radiation dose calculation dose to calculate dose distribution for the Perfexion. Next the optimization will be modified so that the binary variables representing the Gamma Knife C four collimators will be changed to a binary representation of the three available collimator sizes for the eight sectors that comprise the Perfexion collimator. All other aspects of the code should remain the same. The Perfexion planning optimization presents a good future work application of the work developed in this dissertation.

Reference

- [1] Perez, C., Brady, L., *Principles and Practice of Radiation Oncology*, 5th ed. Philadelphia: Lippincott Williams & Wilkins, 2004.
- [2] Khan, F., *The Physics of Radiation Therapy*, 3rd ed. Philadelphia: Lippincott Williams & Wilkins, 2003.
- [3] National Cancer Institute, *Radiation Therapy and You*, NIH Publication No 03-2227, 2003.
- [4] Ganz, J., *Gamma Knife Surgery*, 2nd, revised ed. Wien; New York: Springer, 2004.
- [5] Hall, E., Giaccia, A., *Radiobiology for the Radiologist*, 5th ed. Philadelphia: Lippincott Williams & Wilkins, 2003.
- [6] Bentel, G., *Radiation Therapy Planning*, 2nd ed. Kingsport: McGraw-Hill, 1996.
- [7] Hauge, D., “Baptist Regional Gamma Knife Center Annual Report 2005-2006,” Baptist Regional Gamma Knife, Knoxville, TN, 2006.
- [8] Giller, C., Fielder, J., Gagnon, G., Paddick, I., *Radiosurgical Planning: Gamma Tricks and Cyber Picks*, Hoboken New Jersey: John Wiley & Sons, 2009.
- [9] Elekta., “Leksell Gamma Knife C Instructions for Use,” Elekta, Stockholm, Sweden, Doc. Art. No. 014102, Rev. 00 (06.03).
- [10] Shaw, E., Kline, R., Gillin, M., et al., “Radiation Therapy Oncology Group: radiosurgery quality assurance guidelines,” *International Journal of Radiation Oncology, Biology, & Physics*, vol 27, pp. 1231-1239, 1993.
- [11] Schell, M., Bova, F., Larson, D., Leavitt, D., Latz, W., Podgorsak, E. Wu, A., *Stereotactic Radiosurgery*. College Park, Md: American Association of Physicists in Medicine; AAPM Report 54; Task group 42; 1995.

- [12] Kondziolka, D., *Radiosurgery 1995: 2nd International Stereotactic Radiosurgery Society Meeting*, Switzerland S. Karger AG, 1996.
- [13] International Radiosurgery Society Association. "Stereotactic Radiosurgery Overview," Internet: <http://www.irs.org/radiosurgery.html>, [October 27, 2009]
- [14] Lars Leksell Center for Gamma Knife Surgery. "The Gamma Knife: A Technical Overview." Internet: <http://www.healthsystem.virginia.edu/internet/gammaknife/overview.cfm>, December 6, 2007 [October 27, 2009].
- [15] Elekta., "Site Planning Guide Leksell Gamma Knife C," Elekta, Stockholm, Sweden.
- [16] Yan, Y., Shu, H., Bao, X., Luo, L., Bai, Y., "Clinical treatment planning optimization by Powell's method for gamma unit treatment system," *International Journal of Radiation Oncology, Biology, & Physics*, vol. 39, pp. 247-254, 1997.
- [17] Shu, H., Yan, Y., Luo, L., and Bao, X., "Three-dimensional optimization of treatment planning for gamma unit treatment system," *Medical Physics*, vol. 25, pp. 2352-2357, 1998.
- [18] Gibon, D., Rousseau, J., "Improved Gamma Knife Radiosurgery Using Combinations of Different-Sized Collimators," *Journal of Radiosurgery*, vol. 01, pp. 43-49, 1998.
- [19] Luo, L., Shu, H., Yu, W., Yan, Y., Bao, X., Fu, Y., "Optimizing computerized treatment planning for the gamma knife by source culling," *International Journal of Radiation Oncology, Biology, & Physics*, vol. 45, pp. 1339-1346, 1999.
- [20] Lim, G., Lee, E., *Optimization in Medicine and Biology*, Boca Raton, FL: Auerbach Publications, 2008, pp. 210.

- [21] Leichtman, G., Aita, A., Goldman, W., "Automated Gamma Knife dose planning using polygon clipping and adaptive simulated annealing," *Medical Physics*, vol. 27, pp. 154-162, 2000.
- [22] Zhang, P., Dean, D., Metzger, A., Sibata, C., "Optimization of Gamma Knife treatment planning via guided evolutionary simulated annealing," *Medical Physics*, vol. 28, pp. 1746-1752, 2001.
- [23] Shu, H., Yan, Y., Bao, X., Fu, Y., Luo, L., "Treatment planning optimization by quasi-Newton and simulated annealing methods for gamma unit treatment system," *Physics in Medicine and Biology*, vol. 43, pp. 2795-2805, 1998.
- [24] Ove, R., Popple, R., "Sequential annealing-gradient Gamma Knife radiosurgery optimization," *Physics in Medicine and Biology*, vol. 48, pp. 2071-2080, 2003.
- [25] Wolsey, L., Nemhauser, G., *Integer and Combinatorial Optimization*, New York: Wiley-Interscience, 1999.
- [26] Ferris, M., Shepard, D., *Optimization of Gamma Knife radio- surgery, in Discrete Mathematical Problems with Medical Applications*, D.-Z. Du, Pardalos, P., Wang, J., eds., vol. 55 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 2000, pp. 27-44.
- [27] Lee, E., Fox, T., Crocker, I., Optimization of radiosurgery treatment planning via mixed integer programming, *Medical Physics*, vol. 27, pp. 995-1004, 2000.
- [28] Ruszczynski, A., *Non-linear Optimization*, Princeton New Jersey: Princeton University Press, 2006.
- [29] Shepard, D., Ferris, M., Ove, R., Ma, L., "Inverse treatment planning for Gamma Knife radiosurgery," *Medical Physics*, vol. 27, pp. 2748-2756, 2000.

- [30] Shepard, D., Chin, L., DiBiase, S., Naqvi, S., Lim, J., Ferris, M., "Clinical implementation of an automated planning system for gamma knife radiosurgery," *International Journal of Radiation Oncology, Biology, & Physics*, vol. 56, pp. 1488-1494, 2003.
- [31] Ferris, M., Lim, J., and Shepard, D., "Radiosurgery Treatment Planning via Nonlinear Programming," *Annals of Operations Research*, vol. 119, pp. 247-260, 2003.
- [32] Ferris, M., Lim, J., and Shepard, D., "An Optimization Approach for Radiosurgery Treatment Planning," *SIAM Journal on Optimization*, vol. 13, pp. 921-937, 2003.
- [33] Shepard, D., Ferris, M., Ma, L., and Yu, C., "Benefits of inverse planning as applied to Gamma Knife radiosurgery," presented at Engineering in Medicine and Biology Society, 2000. Proceedings of the 22nd Annual International Conference of the IEEE, 2000.
- [34] Shepard, D., Lim, J., Ferris, M., "Clinical implementation of inverse planning for gamma knife radiosurgery," *International Journal of Radiation Oncology, Biology, & Physics*, vol. 51, pp. 126-127, 2001.
- [35] Shepard, D., Chin, L., DiBiase, D., Naqvi, S., Lim, J., Ferris, M., "Clinical implementation of an automated planning system for gamma knife radiosurgery," *International Journal of Radiation Oncology, Biology, & Physics*, vol. 56, pp. 1488-1494, 2003.
- [36] Ferris, M. Voelker, M., "Fractionation in radiation treatment planning," *Mathematical Programming*, vol. 101, pp. 387 - 413, 2004.

- [37] Bourland, J., Wu, W., "Use of Shape for Automated, Optimized 3D Radiosurgical Treatment Planning," presented at Proceedings of the 4th International Conference on Visualization in Biomedical Computing, 1996.
- [38] Seungjohn, O., Tae-Suk, S., Joo-Young, S., Bo-Young, C., Hyoun-Koo, L., Moon-Chan, K., Taekyu, L., "Development of a rapid planning technique based on heuristic target shaping for stereotactic radiosurgery," *Medical Physics*, vol. 31, pp. 175-182, 2004.
- [39] Wu, Q., Chankong, V., Jitprapaikularn, S., Wessels, B., Einstein, D., Mathayomchann, B., Kinsella, T., "Real-time inverse planning for Gamma Knife radiosurgery," *Medical Physics*, vol. 30, pp. 2988-2995, 2003.
- [40] Wu, Q., Bourland, J., "Morphology-guided radiosurgery treatment planning and optimization for multiple isocenters," *Medical Physics*, vol. 26, pp. 2151-2160, 1999.
- [41] Wu, Q., Bourland, J., "Three-dimensional skeletonization for computer-assisted treatment planning in radiosurgery," *Computerized Medical Imaging and Graphics*, vol. 24, pp. 243-251, 2000.
- [42] Wu, Q., Bourland, J., "A Study and Automatic Solution for Multishot Treatment Planning for the Gamma Knife," *Journal of Radiosurgery*, vol. 3, pp. 77-84, 2000.
- [43] Bourland, J., Wu, Q., "Radiotherapy treatment using medial axis transformation," in *United States Patent*, 2001.
- [44] Wu, Q., "Treatment Planning Optimization for Gamma Knife Radiosurgery," in *The Mayo Graduate School*. Rochester, Minnesota: Mayo Clinic College of Medicine, 1996.
- [45] Ferris, M., Lim, J., Shepard, D., "An Optimization Approach for Radiosurgery Treatment Planning," *SIAM Journal on Optimization*, vol. 13, pp. 921-937, 2003.

- [46] Lim, J., "Optimization in radiation treatment planning," in *Industrial Engineering*. Madison: University of Wisconsin, 2002.
- [47] Wang, J., "Packing of Unequal Spheres and Automated Radiosurgical Treatment Planning," *Journal of Combinatorial Optimization*, vol. 3, pp. 453-463, 1999.
- [48] Wang, J., "Medial Axis and Optimal Locations for Min-Max Sphere Packing," *Journal of Combinatorial Optimization*, vol. 4, pp. 487-503, 2000.
- [49] Lee, K., Barber, D., Walton, L., "Automated gamma knife radiosurgery treatment planning with image registration, data mining, and Nelder-Mead simplex optimization," *Medical Physics*, vol. 33, pp. 2532-2540, 2006.
- [50] Haupt, R., Haupt, S., *Practical Genetic Algorithms*, 2nd ed. Hoboken, New Jersey: John Wiley & Sons, 2004.
- [51] Holland, J., *Adaptation in Natural and Artificial Systems*, The University of Michigan, Ann Arbor, Michigan, 1975.
- [52] Holland, J., "Genetic algorithm," *Scientific American (Int. Ed)*, vol. 267, pp. 66–72, 1992.
- [53] Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, Massachusetts: Addison-Wesley, 1989.
- [54] Guozhen, Y., Reinstein, L., Zhigang, X., "A new genetic algorithm technique in optimization of permanent ¹²⁵I prostate implants," *Medical Physics*, vol. 25, pp. 2308-2315, 1998.
- [55] Gayou O., Das S., Zhou S., Marks L., Parda D., Miften M., "A genetic algorithm for variable selection in logistic regression analysis of radiotherapy treatment outcomes," *Medical Physics*, vol. 35, pp. 5426-5433, 2008.

- [56] Lei J., Li Y., “An approaching genetic algorithm for automatic beam angle selection in IMRT planning,” *Computational Methods Programs Biomedicine*, vol. 93, 2009, pp. 257-265.
- [57] Desai D., Ramsey C., Breinig M., Mahan S., “A topographic leaf-sequencing algorithm for delivering intensity modulated radiation therapy,” *Medical Physics*, vol. 33, 2006, pp. 2751-2756.
- [58] The Math Works, “MATLAB 7 Getting Started Guide,” The Math Works, Natick, MA, September 2009.
- [59] Deasy, J., Blanco, A., Clark, V., “CERR: a computational environment for radiotherapy research,” *Medical Physics*, vol. 30, 2003, pp. 979-85.
- [60] Vanderplaats, G., *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw-Hill, 1993.
- [61] Kjäll, P., “Physics General,” *Advanced Leksell Gamma Plan Workshop*, Atlanta, GA, 2005.
- [62] Paddick, J., “A simple scoring ratio to index the conformity of radiosurgical treatment plans,” *Journal of Neurosurgery (Suppl 3)*, vol. 93, 2000, pp. 219-222.
- [63] Paddick, J., Lippitz, B., “A simple dose gradient measurement tool to complement the conformity index,” *Journal of Neurosurgery (Suppl)*, vol. 105, 2006, pp. 194-201.
- [64] The Math Works, “Genetic Algorithm and Direct Search Toolbox,” The Math Works, Natick, MA, September 2009.
- [65] The Math Works, “Optimization Toolbox,” The Math Works, Natick, MA, September 2009.

- [66] Wood, B., “Using Non-Parametric Modeling to Improve Forensic Facial Approximation.” Master’s thesis, University of Tennessee, 2009.
- [67] Seibert, B., Ramsey C., Gravey, D., Hines, J., Robinson, B., Outten, S., “Verification of helical tomotherapy delivery using autoassociative kernel regression,” *Medical Physics*, vol. 34, 2007, pp. 3249-3262.
- [68] Low, D., Harms, W., Mutic, S., Purdy, J., “A technique for the quantitative evaluation of dose distributions,” *Medical Physics*, vol. 25(5), 1998, pp. 656-661.
- [69] Jolliffe, I.T., *Principal Component Analysis*, 2nd ed. New York: Springer, 2002.

Appendices

Appendix A: MatLab Code

CalcLeksellTranfromforDICOMImport

```
function [planC] = CalcLeksellTranformforDICOMImport(planC);
%
%[planC] = CalcLeksellTranformforDICOMImport(planC);
%
%Using the fiducials on a single slice calculates the transform from
the
%DICOM coordinate space to the Leksell Coordinate space

%Calculate the transformation for the Leksell coordinate system to the
%DICOM coordinate system
%Coordinates of the fiducials
[Fiducials] = ginput(6);
LeksellCord = [5 160;5 1000;5 40;195 160;195 1000;195 40]/10;
FidDiff = LeksellCord - Fiducials;

ZFidDist = [Fiducials(1,2)-Fiducials(2,2) Fiducials(2,2)-Fiducials(3,2)
Fiducials(4,2)-Fiducials(5,2) Fiducials(5,2)-Fiducials(6,2)];
ZTopDist = mean([ZFidDist(1) ZFidDist(3)]);
ZBottomDist = mean([ZFidDist(2) ZFidDist(4)]);
ZLeksellSlice = 16 - ZTopDist;
prompt = {'Enter the dicom slice Z value:'};dlg_title = 'DICOM Z
Value';num_lines = 1;def = {''};
answer = inputdlg(prompt,dlg_title,num_lines,def);
ZDicomSlice = cell2mat(answer);
ZDicomSlice = str2num(ZDicomSlice);

Check = 12 - (ZTopDist + ZBottomDist)

Transform = [mean(FidDiff(:,1));mean([FidDiff(1,2) FidDiff(3,2)
FidDiff(4,2) FidDiff(6,2)]);ZLeksellSlice - ZDicomSlice(1)];
```


Compared2DB

```
function [SelectedPlans, Similar] = Compared2DB(CurROIData, NumPlanSel);
%[SelectedPlans, Similar] = Compared2DB(CurROIData);
%
%Compares the current ROI to be optimized to the database of prior plan
to
%find the most similar plans. If the number of plans variable is
supplied
%the function will select that number of plans. Otherwise it will
select
%the 8 most similar plans
%
%Joseph Bowling University of TN 2011-09-16

if nargin == 2
    NumPlanSel = NumPlanSel;
else
    NumPlanSel = 8;
end

%Load the database
str = computer;
if strcmp(str, 'PCWIN')
    load('C:\TOGA\CERR 3.3 Beta1\UserDefinedFunctions\TOGA\DB.mat');
elseif strcmp(str, 'MACI64')
    load('/TOGA/CERR 3.3 Beta1/UserDefinedFunctions/TOGA/DB.mat');
end

%Performed the Kernel based comparison
%Band = [1 1 1 1 1 1 1 1 1];
Band = std(DB.ROIData(:, 2:10))/3;
%for i = 1:9
for i = [1 5 7 8 9]
    KernelDist(:, i) = gaussmf(DB.ROIData(:, i+1), [Band(i)
CurROIData(i)]);
end
Similarity = [DB.ROIData(:, 1) sum(KernelDist')'];
Similarity = sortrows(Similarity, [2]);
Similarity = flipdim(Similarity, 1);

%Exclude same plans
a = 0;
for i = 1:length(Similarity)
    if Similarity(i, 2) == 5;
        a = 1;
        Similarity(i, 2) = 0;
    else
        Similarity(i, :) = Similarity(i, :);
    end
end
if a == 1
    Similarity = sortrows(Similarity, [2]);
    Similarity = flipdim(Similarity, 1);
end
```

```
SelectedPlans = Similarity(1:NumPlanSel,1);  
Similar = Similarity(1:NumPlanSel,2);
```

Conform Index

```
function
[Iso_Cov, Cov_RTOG, CI_RTOG, CI_Paddick, CI_Gradient, CI_Karolinska, RingIndex] = ConformIndex(planC, structNum, doseNum, RxIsodose, TV)
%
%
%[Iso_Cov, Cov_RTOG, CI_RTOG, CI_Paddick, CI_Gradient, CI_Karolinska, ErrDose] = ConformIndex(planC, structNum, doseNum, RxIsodose, TV)
%
%planC - CERR plan structure
%structNum - Structure index for conformity indices to be determined
%doseNum - Dose grid index for conformity indices to be determined
%RxIsodose - Prescription Isodose
%
%Iso_Cov - Percent Isodose Coverage (Should be greater than 95%)
%Cov_RTOG - RTOG Coverage Index (Should be greater than 0.9)
%CI_RTOG - RTOG Conformity Index (1 is perfect, should be less than 2)
%CI_Paddick - Ian Paddick Conformity Index (Should be greater than 0.6)
%CI_Gradient - Gradient Index (2 is perfect, should be less than 3)
%CI_Karolinska - Gradient Index (Should be less than 4)
%RingIndex - % of points in the a ring surrounding the target volume
that
%is greater than the prescribed dose
%
%TV - Target Volume
%PIV - Prescription Isodose Volume
%TVPIV - Target Volume within Prescription Isodose Volume
%MDTV - Minimum Dose in Target Volume

%Problem setup
indexS = planC{end};
binWidth = 1; %1 Gray is DVH resolution for plotting
dosgrdxdim = abs(planC{indexS.dose}(1, doseNum).verticalGridInterval);
dosgrdydim = planC{indexS.dose}(1, doseNum).horizontalGridInterval;
dosgrdzdim = planC{indexS.dose}(1, doseNum).depthGridInterval;

%Prescription parameters
Rx = planC{indexS.dose}(1, doseNum).Rx;
RxIso = Rx * RxIsodose;

%Calculate Target Volume (TV)
if nargin == 5
else
    TV = getStructureVol(structNum(1), planC);
end
%Calculate % Prescription Isodose Coverage of Target Volume (Should be
%greater than 95%)
[dosesV, volsV, isError] = getDVH(structNum(1), doseNum, planC);
a = find(dosesV >= RxIso);
Iso_Cov = size(a) / size(dosesV);

%Calculate Prescription Isodose Volume (PIV)
b = find(planC{indexS.dose}(1, doseNum).doseArray >= RxIso);
PIV = length(b) * dosgrdxdim * dosgrdydim * dosgrdzdim;
```

```

%RTOG Conformity Index (Should be between 1.0 and 2.0)
CI_RTog = PIV / TV;

%Calculate Minimum Dose to Target Volume (MDTV)
MDTV = min(dosesV);

%RTOG Coverage Index (Should be greater than 0.9)
Cov_RTog = MDTV / RxIso;

%Calculate Target Volume within Prescription Isodose Volume (TVPIV)
TVPIV = Iso_Cov * TV;

%Ian Paddick Conformity Index (Should be greater than 0.6)
CI_Paddick = (TVPIV^2) / (TV * PIV);

%Calculate 1/2 Prescription Isodose Volume (PIV0.5)
c = find(planC{indexS.dose}(1,doseNum).doseArray>=RxIso/2);
PIV0_5= length(c) * dosgrdxdim * dosgrdydim * dosgrdzdim;

%Gradient Index (ratio of PIV to 1/2 PIV)
CI_Gradient = PIV0_5 / PIV;

%Calculate Volume Receiving 10 Gy (PIV10)
d = find(planC{indexS.dose}(1,doseNum).doseArray>=10);
PIV10 = length(d) * dosgrdxdim * dosgrdydim * dosgrdzdim;

%Kar Conformity Index (ratio of TV to 10 Gy Volume)
CI_Karolinska = PIV10 / TV;

%Calculate Isodose Error (SSE of number of dose calculation points
inside
%the target volume not receiving the Rx + SSE of number of dose
calculation
%points outside the target volume and within the optimization ring
%receiving greater than the Rx normalized to the number of dose
calculation
%points)
if size(structNum,1) >= 2;
    if structNum(2,1) ~= 0
        %e = find(dosesV<RxIso);
        %f = length(e) / length(dosesV);
        %f = sum((RxIso-dosesV(e)).^2)/length(dosesV(e));
        [dosesV, volsV, isError] = getDVH(structNum(2,1), doseNum,
planC);
        e = find(dosesV>RxIso);
        RingIndex = length(e) / length(dosesV);
        %h = sum((dosesV(g)-RxIso).^2)/length(dosesV(g));
    end
end

```

ConformityGUI

```
function varargout = ConformityGUI(varargin)
% CONFORMITYGUI MATLAB code for ConformityGUI.fig
%     CONFORMITYGUI, by itself, creates a new CONFORMITYGUI or raises
the existing
%     singleton*.
%
%     H = CONFORMITYGUI returns the handle to a new CONFORMITYGUI or
the handle to
%     the existing singleton*.
%
%     CONFORMITYGUI('CALLBACK',hObject,eventData,handles,...) calls
the local
%     function named CALLBACK in CONFORMITYGUI.M with the given input
arguments.
%
%     CONFORMITYGUI('Property','Value',...) creates a new
CONFORMITYGUI or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before ConformityGUI_OpeningFcn gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to ConformityGUI_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ConformityGUI

% Last Modified by GUIDE v2.5 05-May-2011 13:44:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @ConformityGUI_OpeningFcn, ...
                  'gui_OutputFcn',  @ConformityGUI_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

% End initialization code - DO NOT EDIT

% --- Executes just before ConformityGUI is made visible.
function ConformityGUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% varargin     command line arguments to ConformityGUI (see VARARGIN)

% Choose default command line output for ConformityGUI
handles.output = hObject;

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
end

%Populate objects in the GUI
setGUIpopupmenuPlan(handles);
setGUIpopupmenuROI(handles);
setGUILabelSlideValue(handles);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ConformityGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ConformityGUI_OutputFcn(hObject, eventdata,
handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

```

```

% --- Executes on selection change in popupmenuPlan1.
function popupmenuPlan1_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuPlan1 contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuPlan1

% --- Executes during object creation, after setting all properties.
function popupmenuPlan1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuROI1.
function popupmenuROI1_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuROI1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuROI1 contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuROI1

% --- Executes during object creation, after setting all properties.
function popupmenuROI1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuROI1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%         get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUIDebugLabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function setGUIpopupmenuPlan(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)
    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
end
set(handles.popupmenuPlan1,'String',plannames);
set(handles.popupmenuPlan2,'String',plannames);

function setGUIpopupmenuROI(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the ROIs names
ROIsize = size(handles.planC{handles.indexS.structures});
for i = 1:ROIsize(2)
    ROInames{i,1} =
handles.planC{handles.indexS.structures}(1,i).structureName;
end
set(handles.popupmenuROI1,'String',ROInames);
set(handles.popupmenuROI2,'String',ROInames);

```



```

function setGUILabelSlideValue(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

slidervalue = get(handles.slider1,'Value');
slidervalue = num2str(slidervalue);
set(handles.text5,'String',slidervalue);

slidervalue = get(handles.slider2,'Value');
slidervalue = num2str(slidervalue);
set(handles.text21,'String',slidervalue);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Get the parameters set for the optimization
indexS = handles.planC{end};
doseindex = get(handles.popupmenuPlan1,'Value');
structNum = get(handles.popupmenuROI1,'Value');
RxIsodose = get(handles.slider1,'Value');
TV = getStructureVol(structNum,handles.planC);

%Calculate the conformity
[Iso_Cov,Cov_RTog,CI_RTog,CI_Paddick,CI_Gradient,CI_Karolinska] =
ConformIndex(handles.planC,structNum,doseindex,RxIsodose,TV);

%Populate the text boxes with the conformity indices
set(handles.IsoCov1,'String',num2str(Iso_Cov*100));
set(handles.RTOGCov1,'String',num2str(Cov_RTog));
set(handles.RTOGConf1,'String',num2str(CI_RTog));
set(handles.Paddick1,'String',num2str(CI_Paddick));
set(handles.Gradient1,'String',num2str(CI_Gradient));
set(handles.Karolin1,'String',num2str(CI_Karolinska));
if Iso_Cov < 0.95
    set(handles.IsoCov1,'ForegroundColor',[1 0 0]);
else
    set(handles.IsoCov1,'ForegroundColor',[0 0 0]);
end
if Cov_RTog < 0.90
    set(handles.RTOGCov1,'ForegroundColor',[1 0 0]);
else
    set(handles.RTOGCov1,'ForegroundColor',[0 0 0]);
end
if CI_RTog > 2
    set(handles.RTOGConf1,'ForegroundColor',[1 0 0]);
else
    set(handles.RTOGConf1,'ForegroundColor',[0 0 0]);
end
if CI_Paddick < 0.6

```

```

        set(handles.Paddick1, 'ForegroundColor', [1 0 0]);
    else
        set(handles.Paddick1, 'ForegroundColor', [0 0 0]);
    end
    if CI_Gradient > 3
        set(handles.Gradient1, 'ForegroundColor', [1 0 0]);
    else
        set(handles.Gradient1, 'ForegroundColor', [0 0 0]);
    end
    if CI_Karolinska > 4
        set(handles.Karolin1, 'ForegroundColor', [1 0 0]);
    else
        set(handles.Karolin1, 'ForegroundColor', [0 0 0]);
    end

    Iso_Cov1 = get(handles.IsoCov1, 'String');
    Iso_Cov2 = get(handles.IsoCov2, 'String');
    if isempty(strmatch(Iso_Cov1, '')) & isempty(strmatch(Iso_Cov2, ''))
        PlanDifference(handles)
    end

    PlotDVH(handles)

% --- Executes on selection change in popupmenuPlan2.
function popupmenuPlan2_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject, 'String')) returns
% popupmenuPlan2 contents as cell array
%          contents{get(hObject, 'Value')} returns selected item from
% popupmenuPlan2

% --- Executes during object creation, after setting all properties.
function popupmenuPlan2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
% called

% Hint: popupmenu controls usually have a white background on Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes on selection change in popupmenuROI2.
function popupmenuROI2_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuROI2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

```

```

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuROI2 contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenuROI2

% --- Executes during object creation, after setting all properties.
function popupmenuROI2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuROI2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%         get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUILabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

%Get the parameters set for the optimization
indexS = handles.planC{end};
doseindex = get(handles.popupmenuPlan2, 'Value');
structNum = get(handles.popupmenuROI2, 'Value');
RxIsodose = get(handles.slider2, 'Value');
TV = getStructureVol(structNum, handles.planC);

%Calculate the conformity
[Iso_Cov, Cov_RTOG, CI_RTOG, CI_Paddick, CI_Gradient, CI_Karolinska] =
ConformIndex(handles.planC, structNum, doseindex, RxIsodose, TV);

%Populate the text boxes with the conformity indices
set(handles.IsoCov2, 'String', num2str(Iso_Cov*100));
set(handles.RTOGCov2, 'String', num2str(Cov_RTOG));
set(handles.RTOGConf2, 'String', num2str(CI_RTOG));
set(handles.Paddick2, 'String', num2str(CI_Paddick));
set(handles.Gradient2, 'String', num2str(CI_Gradient));
set(handles.Karolin2, 'String', num2str(CI_Karolinska));
if Iso_Cov < 0.95
    set(handles.IsoCov2, 'ForegroundColor', [1 0 0]);
else
    set(handles.IsoCov2, 'ForegroundColor', [0 0 0]);
end
if Cov_RTOG < 0.90
    set(handles.RTOGCov2, 'ForegroundColor', [1 0 0]);
else
    set(handles.RTOGCov2, 'ForegroundColor', [0 0 0]);
end
if CI_RTOG > 2
    set(handles.RTOGConf2, 'ForegroundColor', [1 0 0]);
else
    set(handles.RTOGConf2, 'ForegroundColor', [0 0 0]);
end
if CI_Paddick < 0.6
    set(handles.Paddick2, 'ForegroundColor', [1 0 0]);
else
    set(handles.Paddick2, 'ForegroundColor', [0 0 0]);
end
if CI_Gradient > 3
    set(handles.Gradient2, 'ForegroundColor', [1 0 0]);
else
    set(handles.Gradient2, 'ForegroundColor', [0 0 0]);
end
if CI_Karolinska > 4
    set(handles.Karolin2, 'ForegroundColor', [1 0 0]);
else
    set(handles.Karolin2, 'ForegroundColor', [0 0 0]);
end

Iso_Cov1 = get(handles.IsoCov1, 'String');
Iso_Cov2 = get(handles.IsoCov2, 'String');
if isempty(strmatch(Iso_Cov1, '')) & isempty(strmatch(Iso_Cov2, ''))
    PlanDifference(handles);
end

PlotDVH(handles)

```

```

function PlanDifference(handles);
% handles      structure with handles and user data (see GUIDATA)

Iso_Cov1 = get(handles.IsoCov1, 'String');
Iso_Cov2 = get(handles.IsoCov2, 'String');
set(handles.IsoCovDiff, 'String', num2str(str2num(Iso_Cov1) -
str2num(Iso_Cov2)));

Cov_RT0G1 = get(handles.RTOGCov1, 'String');
Cov_RT0G2 = get(handles.RTOGCov2, 'String');
set(handles.RTOGCovDiff, 'String', num2str(str2num(Cov_RT0G1) -
str2num(Cov_RT0G2)));

RTOGConf1 = get(handles.RTOGConf1, 'String');
RTOGConf2 = get(handles.RTOGConf2, 'String');
set(handles.RTOGConfDiff, 'String', num2str(str2num(RTOGConf1) -
str2num(RTOGConf2)));

Paddick1 = get(handles.Paddick1, 'String');
Paddick2 = get(handles.Paddick2, 'String');
set(handles.PaddickDiff, 'String', num2str(str2num(Paddick1) -
str2num(Paddick2)));

Gradient1 = get(handles.Gradient1, 'String');
Gradient2 = get(handles.Gradient2, 'String');
set(handles.GradientDiff, 'String', num2str(str2num(Gradient1) -
str2num(Gradient2)));

Karolin1 = get(handles.Karolin1, 'String');
Karolin2 = get(handles.Karolin2, 'String');
set(handles.KarolinDiff, 'String', num2str(str2num(Karolin1) -
str2num(Karolin2)));

if str2num(Iso_Cov1) > str2num(Iso_Cov2)
    set(handles.IsoCovDiff, 'ForegroundColor', [0 0 1]);
elseif str2num(Iso_Cov1) < str2num(Iso_Cov2)
    set(handles.IsoCovDiff, 'ForegroundColor', [0 0.4 0.2]);
end
if str2num(Cov_RT0G1) > str2num(Cov_RT0G2)
    set(handles.RTOGCovDiff, 'ForegroundColor', [0 0 1]);
elseif str2num(Cov_RT0G1) < str2num(Cov_RT0G2)
    set(handles.RTOGCovDiff, 'ForegroundColor', [0 0.4 0.2]);
end
if str2num(RTOGConf1) < str2num(RTOGConf2)
    set(handles.RTOGConfDiff, 'ForegroundColor', [0 0 1]);
elseif str2num(RTOGConf1) > str2num(RTOGConf2)
    set(handles.RTOGConfDiff, 'ForegroundColor', [0 0.4 0.2]);
end
if str2num(Paddick1) > str2num(Paddick2)
    set(handles.PaddickDiff, 'ForegroundColor', [0 0 1]);
elseif str2num(Paddick1) < str2num(Paddick2)
    set(handles.PaddickDiff, 'ForegroundColor', [0 0.4 0.2]);
end
if str2num(Gradient1) < str2num(Gradient2)

```

```

        set(handles.GradientDiff, 'ForegroundColor', [0 0 1]);
elseif str2num(Gradient1) > str2num(Gradient2)
    set(handles.GradientDiff, 'ForegroundColor', [0 0.4 0.2]);
end
if str2num(Karolin1) < str2num(Karolin2)
    set(handles.KarolinDiff, 'ForegroundColor', [0 0 1]);
elseif str2num(Karolin1) > str2num(Karolin2)
    set(handles.KarolinDiff, 'ForegroundColor', [0 0.4 0.2]);
end

function PlotDVH(handles);
% handles      structure with handles and user data (see GUIDATA)

%Get the parameters set for the optimization
indexS = handles.planC{end};

Iso_Cov1 = get(handles.IsoCov1, 'String');
if isempty(strmatch(Iso_Cov1, ''))
    doseindex = get(handles.popupmenuPlan1, 'Value');
    structNum = get(handles.popupmenuROI1, 'Value');
    RxIsodose = get(handles.slider1, 'Value');
    TV = getStructureVol(structNum, handles.planC);
    %Calculate the DVH
    [handles.planC, doseBinsV, volsHistV] = getDVHMatrix(handles.planC,
structNum, doseindex);
    [dose, percentVol] = diff2CumulativeDVH(doseBinsV, volsHistV);
    %Plot the DVH
    hold on;
    plot(handles.axes1, dose, percentVol)
    grid on
end

Iso_Cov2 = get(handles.IsoCov2, 'String');
if isempty(strmatch(Iso_Cov2, ''))
    doseindex = get(handles.popupmenuPlan2, 'Value');
    structNum = get(handles.popupmenuROI2, 'Value');
    RxIsodose = get(handles.slider2, 'Value');
    TV = getStructureVol(structNum, handles.planC);
    %Calculate the DVH
    [handles.planC, doseBinsV, volsHistV] = getDVHMatrix(handles.planC,
structNum, doseindex);
    [dose, percentVol] = diff2CumulativeDVH(doseBinsV, volsHistV);
    %Plot the DVH
    hold on;
    plot(handles.axes1, dose, percentVol, 'Color', [0 0.4
0.2], 'LineStyle', ':', 'LineWidth', 3)
    grid on
end

```

Convert2CERR

```
function Convert2Cerr(directory);
%Funtion that converts treatment planning data from GammaPlan treatment
%planning system into a CERR importable format.
%
%   Convert2Cerr(directory);
%   directory - directory to convert from GammaPlan data to CERR
importable
%
%Joseph Bowling
%University of Tennessee
%11-26-2007

%Select directory of files
n = nargin;
if n == 0
    directory = uigetdir;
end
if directory == 0
    return
end

%Original Directory
OrgDir = pwd;

%Change directory
cd(directory)

%Determine what is in the current directory
current_folder = dir;

%Check if CERR directory
count = 0;
for i = 1:length(current_folder)
    if ~isempty(strmatch(current_folder(i).name, strvcat('Fusions',
'Logs', 'Orientations', 'Shots', 'Sortings', 'SRdata'),'exact')));
        count = count + 1;
    end
end
if count < 5
    error('Not a CERR directory')
    finish
end

>Delete all files (anything that isn't a directory)
path = pwd;
for i = 1:length(current_folder)
    if ~current_folder(i).isdir
        filename = [path '\\' current_folder(i).name];
        delete(filename)
    end
end

%Rename DoseGroup
```

```

path = [pwd '\SRData'];
cd(path);
DataDir = dir;
count = 0;
for i = 1:length(DataDir)
    if ~isempty(strmatch('DoseGroup', strvcats(DataDir(i).name)));
        if ~isempty(findstr(DataDir(i).name, 'A. ')) |
~isempty(findstr(DataDir(i).name, 'B. ')) |
~isempty(findstr(DataDir(i).name, 'C. ')) |
~isempty(findstr(DataDir(i).name, 'D. ')) |
~isempty(findstr(DataDir(i).name, 'F. '))
            name = ['DoseGroup' num2str(count) '.1'];
            copyfile(DataDir(i).name, name);
            delete(DataDir(i).name);
            count = count + 1;
        end
    end
end
cd(directory);

%Rename Shots
path = [pwd '\Shots'];
cd(path);
DataDir = dir;
count = 0;
for i = 1:length(DataDir)
    if ~isempty(strmatch('Shot', strvcats(DataDir(i).name)));
        if ~isempty(findstr(DataDir(i).name, 'A. ')) |
~isempty(findstr(DataDir(i).name, 'B. ')) |
~isempty(findstr(DataDir(i).name, 'C. ')) |
~isempty(findstr(DataDir(i).name, 'D. ')) |
~isempty(findstr(DataDir(i).name, 'F. '))
            name = ['Shot.1'];
            copyfile(DataDir(i).name, name);
            delete(DataDir(i).name);
            count = count + 1;
        end
    end
end
cd(directory);

%Convert back to original directory
cd(OrgDir);

%Save to TOGA planning database
%Add code here

```


CreatePriorPlanDBEntry

```
function [DB] = CreatePriorPlanDBEntry(structNum, planNum, DB, planC)

%Initialize the function
global planC
indexS = planC{end};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Calculate the comparison parameters for the selected structure
output = structureStats(structNum, planC);

%Get the strucute volume
TV = output.vol;

%Get the largest dimension in X, Y, Z of the structure
LargeDim = output.max - output.min;

%Get the contour points
numslices = length(planC{indexS.structures}(1,structNum).contour);
structPoints = [];
for i = 1:numslices;
    if
~isempty(planC{indexS.structures}(1,structNum(1)).contour(1,i).segments
.points)
        temp =
planC{indexS.structures}(1,structNum).contour(1,i).segments.points;
        tempsize = size(temp,1);
        strPsize = size(structPoints,1);
        structPoints(strPsize+1:strPsize+tempsize,:) = [temp
repmat(i,tempsize,1)];
    end
end
structPoints(:,3) = structPoints(:,3) +
planC{indexS.scan}(1,1).transM(3,4);

%Get the center of mass of the structure
%COM = output.COM;
COM = mean(structPoints(:,[1 2 3]));

%Get the raster segments for the structure
[structRasterSegments] = getRasterSegments(structNum, planC);

%Calculate the spread of the structure from the center of mass
structDistCOM = (sum((structPoints(:,[1 2 3]) -
repmat(COM,size(structPoints,1),1)).^2,2)).^(1/2);
structDistMeanCOM = mean(structDistCOM);
structDistVarCOM = var(structDistCOM);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%PCA analysis
%Perform the PCA
```

```

[coeff,score,roots,t2] = princomp(structPoints(:,1:3));
pctExplained = roots' ./ sum(roots);

PC = 2.*sqrt(2.*roots);

%Plot the data with the PCA
figure(10);plot3(structPoints(:,1),structPoints(:,2),structPoints(:,3),
'bo','MarkerSize',5);
grid on;

%Add PCA arrow to graph
LinePoint(1,:) = sqrt(2.*roots(1))*coeff(:,1)';
LinePoint(2,:) = sqrt(2.*roots(2))*coeff(:,2)';
LinePoint(3,:) = sqrt(2.*roots(3))*coeff(:,3)';
LinePoint(1,:) = LinePoint(1,:) + COM;
LinePoint(2,:) = LinePoint(2,:) + COM;
LinePoint(3,:) = LinePoint(3,:) + COM;
line([COM(1) LinePoint(1,1)], [COM(2) LinePoint(1,2)], [COM(3)
LinePoint(1,3)], 'Color','r', 'LineWidth',5);
line([COM(1) LinePoint(2,1)], [COM(2) LinePoint(2,2)], [COM(3)
LinePoint(2,3)], 'Color','g', 'LineWidth',5);
line([COM(1) LinePoint(3,1)], [COM(2) LinePoint(3,2)], [COM(3)
LinePoint(3,3)], 'Color','m', 'LineWidth',5)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%Get the shot data for the selected plan
shotstemp= planC{indexS.beamGeometry}(1,planNum).shots;
for i = 1:length(shotstemp)
    SelectShots(i,1) = shotstemp(i).collimator_helmet;
    SelectShots(i,2) = shotstemp(i).x;
    SelectShots(i,3) = shotstemp(i).y;
    SelectShots(i,4) = shotstemp(i).z;
    SelectShots(i,5) = shotstemp(i).weight;
    SelectShots(i,6) = shotstemp(i).gammaTOGA;
end
Shots(:,3:8) = SelectShots;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%Add the plan to the database
index = size(DB.ROIData,1)+1;
Shots(:,2) = repmat(index,size(Shots,1),1);
Shots(:,2) = repmat(index,size(Shots,1),1);
if length(DB.Shots) == 0
    holder = 0;
else
    holder = DB.Shots(end,1);
end
Shots(:,1) = (holder+1:holder+size(Shots,1))';
DB.Shots = [DB.Shots;Shots];

```

```

DB.Structure{1} = 'Plan Key';
DB.Structure{2} = 'Tumor Volume';
DB.Structure{3} = 'Large Dimension X';
DB.Structure{4} = 'Large Dimension Y';
DB.Structure{5} = 'Large Dimension Z';
DB.Structure{6} = 'ROI points mean distance from center of mass';
DB.Structure{7} = 'Variance of ROI points distance from center of
mass';
DB.Structure{8} = 'Principal comp 1';
DB.Structure{9} = 'Principal comp 2';
DB.Structure{10} = 'Principal comp 3';

DB.Name{index,1} = index;
DB.Name{index,2} = output.name;
DB.Name{index,3} = planC{indexS.dose}(1,planNum).Rx;

DB.ROIData(index,1) = index;
DB.ROIData(index,2) = TV;
DB.ROIData(index,[3 4 5]) = LargeDim;
DB.ROIData(index,6) = structDistMeanCOM;
DB.ROIData(index,7) = structDistVarCOM;
DB.ROIData(index,[8 9 10]) = PC;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%Save the DB variable
savefile = 'C:\TOGA\CERR 3.3 Beta1\UserDefinedFunctions\TOGA\DB.mat';
save(savefile, 'DB');

```

CrossOverScatteredTOGA

```
function xoverKids =  
crossoverscatteredTOGA(parents,options,GenomeLength,FitnessFcn,unused,t  
hisPopulation)  
%CROSSOVERSCATTERED Position independent crossover function.  
%   XOVERKIDS = CROSSOVERSCATTERED(PARENTS,OPTIONS,GENOMELENGTH, ...  
%   FITNESSFCN,SCORES,THISPOPULATION) creates the crossover children  
XOVERKIDS  
%   of the given population THISPOPULATION using the available PARENTS.  
%   In single or double point crossover, genomes that are near each  
other tend  
%   to survive together, whereas genomes that are far apart tend to be  
%   separated. The technique used here eliminates that effect. Each  
gene has an  
%   equal chance of coming from either parent. This is sometimes called  
uniform  
%   or random crossover.  
%  
%   Example:  
%   Create an options structure using CROSSOVERSCATTERED as the  
crossover  
%   function  
%   options = gaoptimset('CrossoverFcn',@crossoverscattered);  
  
%   Copyright 2003-2007 The MathWorks, Inc.  
%   $Revision: 1.1.6.1 $   $Date: 2009/08/29 08:24:20 $  
  
% How many children to produce?  
nKids = length(parents)/2;  
% Extract information about linear constraints, if any  
%linCon = options.LinearConstr;  
%constr = ~isequal(linCon.type,'unconstrained');  
% Allocate space for the kids  
xoverKids = zeros(nKids,GenomeLength);  
  
% To move through the parents twice as fast as the kids are  
% being produced, a separate index for the parents is needed  
index = 1;  
% for each kid...  
for i=1:nKids  
    % get parents  
    r1 = parents(index);  
    index = index + 1;  
    r2 = parents(index);  
    index = index + 1;  
    % Randomly select half of the genes from each parent  
    % This loop may seem like brute force, but it is twice as fast as  
the  
    % vectorized version, because it does no allocation.  
    for j = 1:GenomeLength  
        if(rand > 0.5)  
            xoverKids(i,j) = thisPopulation(r1,j);  
        else
```

```

        xoverKids(i,j) = thisPopulation(r2,j);
    end
end
% Make sure that offspring are feasible w.r.t. linear constraints
%if constr
%    feasible =
isTrialFeasible(xoverKids(i,:) ',linCon.Aineq,linCon.bineq,linCon.Aeq,
...
%        linCon.beq,linCon.lb,linCon.ub,sqrt(options.TolCon));
%    if ~feasible % Kid is not feasible
%        % Children are arithmetic mean of two parents (feasible
w.r.t
%        % linear constraints)
%        alpha = rand;
%        xoverKids(i,:) = alpha*thisPopulation(r1,:) + ...
%            (1-alpha)*thisPopulation(r2,:);
%    end
%end
end
%xoverKids

```

DoseCalcGUI

```
function varargout = DoseCalcGUI(varargin)
% DOSECALCGUI M-file for DoseCalcGUI.fig
%     DOSECALCGUI, by itself, creates a new DOSECALCGUI or raises the
existing
%     singleton*.
%
%     H = DOSECALCGUI returns the handle to a new DOSECALCGUI or the
handle to
%     the existing singleton*.
%
%     DOSECALCGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in DOSECALCGUI.M with the given input
arguments.
%
%     DOSECALCGUI('Property','Value',...) creates a new DOSECALCGUI or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before DoseCalcGUI_OpeningFunction gets
called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to DoseCalcGUI_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help DoseCalcGUI

% Last Modified by GUIDE v2.5 16-Nov-2010 20:51:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @DoseCalcGUI_OpeningFcn, ...
                  'gui_OutputFcn',  @DoseCalcGUI_OutputFcn, ...
                  'gui_LayoutFcn',   @DoseCalcGUI_LayoutFcn, ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

% End initialization code - DO NOT EDIT

% --- Executes just before DoseCalcGUI is made visible.
function DoseCalcGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to DoseCalcGUI (see VARARGIN)

% Choose default command line output for DoseCalcGUI
handles.output = hObject;

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
end

%Populate objects in the GUI
setGUIpopupmenuPlan(handles);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DoseCalcGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = DoseCalcGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenuPlan.
function popupmenuPlan_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuPlan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

```

```

% Hints: contents = get(hObject,'String') returns popupmenuPlan
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenuPlan

% --- Executes during object creation, after setting all properties.
function popupmenuPlan_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuPlan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbuttonCalcDose.
function pushbuttonCalcDose_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonCalcDose (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get the selected plan
planindex = get(handles.popupmenuPlan, 'Value');

%Get the calculation method
calcWeight = get(handles.radiobuttonWeight, 'Value');
calcTime = get(handles.radiobuttonTime, 'Value');

if calcWeight == 1
    CalcMethod = 1;
elseif calcTime == 1
    CalcMethod = 0;
end

[Profiles] = loadprofilesTOGA(handles.planC, planindex);

%Calculate the dose
[handles.planC] =
dosecalcTOGA(handles.planC, planindex, Profiles, CalcMethod);

%Update planC with the new dose matrix position
assignin('base', 'planC', handles.planC);
guidata(hObject, handles);

function setGUIpopupmenuPlan(handles)
% handles    structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices

```



```

%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)
    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
end
set(handles.popupmenuPlan, 'String', plannames);

% --- Executes on button press in checkboxWeight.
function checkboxWeight_Callback(hObject, eventdata, handles)
% hObject      handle to checkboxWeight (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxWeight

% --- Executes on button press in checkboxTime.
function checkboxTime_Callback(hObject, eventdata, handles)
% hObject      handle to checkboxTime (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxTime

% --- Creates and returns a handle to the GUI figure.
function h1 = DoseCalcGUI_LayoutFcn(policy)
% policy - create a new figure or use a singleton. 'new' or 'reuse'.

persistent hsingleton;
if strcmpi(policy, 'reuse') & ishandle(hsingleton)
    h1 = hsingleton;
    return;
end
load DoseCalcGUI.mat

appdata = [];
appdata.GUIDEOptions = struct(...
    'active_h', [], ...
    'taginfo', struct(...
    'figure', 2, ...
    'popupmenu', 2, ...
    'pushbutton', 2, ...
    'checkbox', 3, ...
    'radiobutton', 4, ...
    'uipanel', 2), ...
    'override', 0, ...
    'release', 13, ...

```

```

        'resize', 'none', ...
        'accessibility', 'callback', ...
        'mfile', 1, ...
        'callbacks', 1, ...
        'singleton', 1, ...
        'syscolorfig', 1, ...
        'blocking', 0, ...
        'lastSavedFile', '/Users/jbowling/Documents/MATLAB/DoseCalcGUI.m',
    ...
        'lastFilename', '/TOGA/CERR 3.3
Beta1/UserDefinedFunctions/TOGA/DoseCalcGUI.fig');
appdata.lastValidTag = 'figure1';
appdata.GUIDELayoutEditor = [];
appdata.initTags = struct(...
    'handle', [], ...
    'tag', 'figure1');

h1 = figure(...
    'Units','characters',...
    'Color',[0.701960784313725 0.701960784313725 0.701960784313725],...
    'Colormap',[0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;0 0 0.8125;0 0
0.875;0 0 0.9375;0 0 1;0 0.0625 1;0 0.125 1;0 0.1875 1;0 0.25 1;0
0.3125 1;0 0.375 1;0 0.4375 1;0 0.5 1;0 0.5625 1;0 0.625 1;0 0.6875 1;0
0.75 1;0 0.8125 1;0 0.875 1;0 0.9375 1;0 1 1;0.0625 1 1;0.125 1
0.9375;0.1875 1 0.875;0.25 1 0.8125;0.3125 1 0.75;0.375 1 0.6875;0.4375
1 0.625;0.5 1 0.5625;0.5625 1 0.5;0.625 1 0.4375;0.6875 1 0.375;0.75 1
0.3125;0.8125 1 0.25;0.875 1 0.1875;0.9375 1 0.125;1 1 0.0625;1 1 0;1
0.9375 0;1 0.875 0;1 0.8125 0;1 0.75 0;1 0.6875 0;1 0.625 0;1 0.5625
0;1 0.5 0;1 0.4375 0;1 0.375 0;1 0.3125 0;1 0.25 0;1 0.1875 0;1 0.125
0;1 0.0625 0;1 0 0;0.9375 0 0;0.875 0 0;0.8125 0 0;0.75 0 0;0.6875 0
0;0.625 0 0;0.5625 0 0],...
    'IntegerHandle','off',...
    'InvertHardcopy',get(0,'defaultfigureInvertHardcopy'),...
    'KeyPressFcn','CERRHotKeys',...
    'MenuBar','none',...
    'Name','DoseCalcGUI',...
    'NumberTitle','off',...
    'PaperPosition',get(0,'defaultfigurePaperPosition'),...
    'Position',[103.714285714286 49.8571428571429 60.2857142857143
11.6428571428571],...
    'Resize','off',...
    'CreateFcn',{@local_CreateFcn,
    'set(gcbo,'KeyPressFcn','CERRHotKeys'),' , appdata} ,...
    'HandleVisibility','callback',...
    'Tag','figure1',...
    'UserData',[],...
    'Visible','on');

appdata = [];
appdata.lastValidTag = 'popupmenuPlan';

h2 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'BackgroundColor',[1 1 1],...

```

```

'Callback',@(hObject,eventdata)DoseCalcGUI('popupmenuPlan_Callback',hOb
ject,eventdata,guidata(hObject)),...
'Position',[5.2 8.84615384615385 26.8 1.53846153846154],...
'String','Plan-',...
'Style','popupmenu',...
'Value',1,...
'CreateFcn',{@local_CreateFcn,
@(hObject,eventdata)DoseCalcGUI('popupmenuPlan_CreateFcn',hObject,event
data,guidata(hObject)), appdata} ,...
'Tag','popupmenuPlan');

appdata = [];
appdata.lastValidTag = 'pushbuttonCalcDose';

h3 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback',@(hObject,eventdata)DoseCalcGUI('pushbuttonCalcDose_Callback
',hObject,eventdata,guidata(hObject)),...
'Position',[38.2 8.61538461538461 13.8 1.76923076923077],...
'String','Calc Dose',...
'Tag','pushbuttonCalcDose',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'uipanel1';

h4 = uibuttongroup(...
'Parent',h1,...
'Title','Dose Calc Method',...
'Tag','uipanel1',...
'Clipping','on',...
'Position',[0.0930232558139535 0.112582781456954 0.451827242524917
0.596026490066225],...
'SelectedObject',[],...
'SelectionChangeFcn',[],...
'OldSelectedObject',[],...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'radiobuttonWeight';

h5 = uicontrol(...
'Parent',h4,...
'Units','normalized',...
'Callback',mat{1},...
'Position',[0.0610687022900763 0.460526315789474 0.9 0.2],...
'String','Weight',...
'Style','radiobutton',...
'Value',1,...
'Tag','radiobuttonWeight',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'radiobuttonTime';

```

```

h6 = uicontrol(...
    'Parent',h4,...
    'Units','normalized',...
    'Callback',mat{2},...
    'Position',[0.0610687022900763 0.131578947368421 0.9 0.2],...
    'String','Time',...
    'Style','radiobutton',...
    'Tag','radiobuttonTime',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

hsingleton = h1;

% --- Set application data first then calling the CreateFcn.
function local_CreateFcn(hObject, eventdata, createfcn, appdata)

if ~isempty(appdata)
    names = fieldnames(appdata);
    for i=1:length(names)
        name = char(names(i));
        setappdata(hObject, name, getfield(appdata,name));
    end
end

if ~isempty(createfcn)
    if isa(createfcn,'function_handle')
        createfcn(hObject, eventdata);
    else
        eval(createfcn);
    end
end

% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)

gui_StateFields = {'gui_Name'
    'gui_Singleton'
    'gui_OpeningFcn'
    'gui_OutputFcn'
    'gui_LayoutFcn'
    'gui_Callback'};
gui_Mfile = '';
for i=1:length(gui_StateFields)
    if ~isfield(gui_State, gui_StateFields{i})
        error('MATLAB:gui_mainfcn:FieldNotFound', 'Could not find field
%s in the gui_State struct in GUI M-file %s', gui_StateFields{i},
gui_Mfile);
    elseif isequal(gui_StateFields{i}, 'gui_Name')
        gui_Mfile = [gui_State.(gui_StateFields{i}), '.m'];
    end
end
end

```

```

numargin = length(varargin);

if numargin == 0
    % DOSECALCGUI
    % create the GUI only if we are not in the process of loading it
    % already
    gui_Create = true;
elseif local_isInvokeActiveXCallback(gui_State, varargin{:})
    % DOSECALCGUI(ACTIVEX,...)
    vin{1} = gui_State.gui_Name;
    vin{2} = [get(varargin{1}.Peer, 'Tag'), '_', varargin{end}];
    vin{3} = varargin{1};
    vin{4} = varargin{end-1};
    vin{5} = guidata(varargin{1}.Peer);
    feval(vin{:});
    return;
elseif local_isInvokeHGCallback(gui_State, varargin{:})
    % DOSECALCGUI('CALLBACK', hObject, eventData, handles,...)
    gui_Create = false;
else
    % DOSECALCGUI(...)
    % create the GUI and hand varargin to the openingfcn
    gui_Create = true;
end

if ~gui_Create
    % In design time, we need to mark all components possibly created
    in
    % the coming callback evaluation as non-serializable. This way,
    they
    % will not be brought into GUIDE and not be saved in the figure
    file
    % when running/saving the GUI from GUIDE.
    designEval = false;
    if (numargin>1 && ishghandle(varargin{2}))
        fig = varargin{2};
        while ~isempty(fig) && ~isa(handle(fig), 'figure')
            fig = get(fig, 'parent');
        end

        designEval = isappdata(0, 'CreatingGUIDEfigure') ||
isprop(fig, '__GUIDEfigure');
    end

    if designEval
        beforeChildren = findall(fig);
    end

    % evaluate the callback now
    varargin{1} = gui_State.gui_Callback;
    if nargin
        [varargout{1:nargout}] = feval(varargin{:});
    else
        feval(varargin{:});
    end
end

```

```

        % Set serializable of objects created in the above callback to off
in
        % design time. Need to check whether figure handle is still valid
in
        % case the figure is deleted during the callback dispatching.
        if designEval && ishandle(fig)
            set(setdiff(findall(fig),beforeChildren),
'Serializable','off');
        end
    else
        if gui_State.gui_Singleton
            gui_SingletonOpt = 'reuse';
        else
            gui_SingletonOpt = 'new';
        end

        % Check user passing 'visible' P/V pair first so that its value can
be
        % used by oepnfig to prevent flickering
        gui_Visible = 'auto';
        gui_VisibleInput = '';
        for index=1:2:length(varargin)
            if length(varargin) == index || ~ischar(varargin{index})
                break;
            end

            % Recognize 'visible' P/V pair
            len1 = min(length('visible'),length(varargin{index}));
            len2 = min(length('off'),length(varargin{index+1}));
            if ischar(varargin{index+1}) &&
strncmpi(varargin{index},'visible',len1) && len2 > 1
                if strncmpi(varargin{index+1},'off',len2)
                    gui_Visible = 'invisible';
                    gui_VisibleInput = 'off';
                elseif strncmpi(varargin{index+1},'on',len2)
                    gui_Visible = 'visible';
                    gui_VisibleInput = 'on';
                end
            end
        end

        % Open fig file with stored settings. Note: This executes all
component
        % specific CreateFunctions with an empty HANDLES structure.

        % Do feval on layout code in m-file if it exists
        gui_Exported = ~isempty(gui_State.gui_LayoutFcn);
        % this application data is used to indicate the running mode of a
GUIDE
        % GUI to distinguish it from the design mode of the GUI in GUIDE.
it is
        % only used by actxproxy at this time.
        setappdata(0,genvarname(['OpenGuiWhenRunning_',
gui_State.gui_Name]),1);
        if gui_Exported

```

```

gui_hFigure = feval(gui_State.gui_LayoutFcn, gui_SingletonOpt);

% make figure invisible here so that the visibility of figure
is
% consistent in OpeningFcn in the exported GUI case
if isempty(gui_VisibleInput)
    gui_VisibleInput = get(gui_hFigure, 'Visible');
end
set(gui_hFigure, 'Visible', 'off')

% openfig (called by local_openfig below) does this for guis
without
% the LayoutFcn. Be sure to do it here so guis show up on
screen.
movegui(gui_hFigure, 'onscreen');
else
    gui_hFigure = local_openfig(gui_State.gui_Name,
gui_SingletonOpt, gui_Visible);
    % If the figure has InGUIInitialization it was not completely
created
    % on the last pass. Delete this handle and try again.
    if isappdata(gui_hFigure, 'InGUIInitialization')
        delete(gui_hFigure);
        gui_hFigure = local_openfig(gui_State.gui_Name,
gui_SingletonOpt, gui_Visible);
    end
    end
    if isappdata(0, genvarname(['OpenGuiWhenRunning_',
gui_State.gui_Name]))
        rmappdata(0, genvarname(['OpenGuiWhenRunning_',
gui_State.gui_Name]));
    end

    % Set flag to indicate starting GUI initialization
    setappdata(gui_hFigure, 'InGUIInitialization', 1);

    % Fetch GUIDE Application options
    gui_Options = getappdata(gui_hFigure, 'GUIDEOptions');
    % Singleton setting in the GUI M-file takes priority if different
    gui_Options.singleton = gui_State.gui_Singleton;

    if ~isappdata(gui_hFigure, 'GUIOnScreen')
        % Adjust background color
        if gui_Options.syscolorfig
            set(gui_hFigure, 'Color',
get(0, 'DefaultUiControlBackgroundColor'));
        end

        % Generate HANDLES structure and store with GUIDATA. If there
is
        % user set GUI data already, keep that also.
        data = guidata(gui_hFigure);
        handles = guihandles(gui_hFigure);
        if ~isempty(handles)
            if isempty(data)
                data = handles;
            end
        end
    end
end

```

```

        else
            names = fieldnames(handles);
            for k=1:length(names)
                data.(char(names(k)))=handles.(char(names(k)));
            end
        end
    end
    guidata(gui_hFigure, data);
end

% Apply input P/V pairs other than 'visible'
for index=1:2:length(varargin)
    if length(varargin) == index || ~ischar(varargin{index})
        break;
    end

    len1 = min(length('visible'),length(varargin{index}));
    if ~strncmpi(varargin{index},'visible',len1)
        try set(gui_hFigure, varargin{index}, varargin{index+1}),
        catch break, end
    end
end

% If handle visibility is set to 'callback', turn it on until
finished
% with OpeningFcn
gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
if strcmp(gui_HandleVisibility, 'callback')
    set(gui_hFigure,'HandleVisibility', 'on');
end

feval(gui_State.gui_OpeningFcn, gui_hFigure, [],
guidata(gui_hFigure), varargin{:});

if isscalar(gui_hFigure) && ishandle(gui_hFigure)
    % Handle the default callbacks of predefined toolbar tools in
this
    % GUI, if any

%guidemfile('restoreToolbarToolPredefinedCallback',gui_hFigure);

    % Update handle visibility
    set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);

    % Call openfig again to pick up the saved visibility or apply
the
    % one passed in from the P/V pairs
    if ~gui_Exported
        gui_hFigure = local_openfig(gui_State.gui_Name,
'reuse',gui_Visible);
    elseif ~isempty(gui_VisibleInput)
        set(gui_hFigure,'Visible',gui_VisibleInput);
    end
    if strcmpi(get(gui_hFigure, 'Visible'), 'on')
        figure(gui_hFigure);
    end
end

```



```

        if gui_Options.singleton
            setappdata(gui_hFigure, 'GUIOnScreen', 1);
        end
    end

    % Done with GUI initialization
    if isappdata(gui_hFigure, 'InGUIInitialization')
        rmappdata(gui_hFigure, 'InGUIInitialization');
    end

    % If handle visibility is set to 'callback', turn it on until
    % finished with OutputFcn
    gui_HandleVisibility = get(gui_hFigure, 'HandleVisibility');
    if strcmp(gui_HandleVisibility, 'callback')
        set(gui_hFigure, 'HandleVisibility', 'on');
    end
    gui_Handles = guidata(gui_hFigure);
else
    gui_Handles = [];
end

if nargout
    [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn,
gui_hFigure, [], gui_Handles);
else
    feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
end

if isscalar(gui_hFigure) && ishandle(gui_hFigure)
    set(gui_hFigure, 'HandleVisibility', gui_HandleVisibility);
end
end

function gui_hFigure = local_openfig(name, singleton, visible)

% openfig with three arguments was new from R13. Try to call that
% first, if
% failed, try the old openfig.
if nargin('openfig') == 2
    % OPENFIG did not accept 3rd input argument until R13,
    % toggle default figure visible to prevent the figure
    % from showing up too soon.
    gui_OldDefaultVisible = get(0, 'defaultFigureVisible');
    set(0, 'defaultFigureVisible', 'off');
    gui_hFigure = openfig(name, singleton);
    set(0, 'defaultFigureVisible', gui_OldDefaultVisible);
else
    gui_hFigure = openfig(name, singleton, visible);
end

function result = local_isInvokeActiveXCallback(gui_State, varargin)

try
    result = ispc && iscom(varargin{1}) ...

```

```

        && isequal(varargin{1},gcbo);
catch
    result = false;
end

function result = local_isInvokeHGCallback(gui_State, varargin)

try
    fhandle = functions(gui_State.gui_Callback);
    result = ~isempty(findstr(gui_State.gui_Name,fhandle.file)) || ...
        (ischar(varargin{1}) ...
        && isequal(ishandle(varargin{2}), 1) ...
        && (~isempty(strfind(varargin{1},[get(varargin{2}, 'Tag'),
'_' ])) || ...
        ~isempty(strfind(varargin{1}, '_CreateFcn'))));
catch
    result = false;
end

```

dosecalcTOGA

```
function [planC] =  
dosecalcTOGA(planC,doseindex,Profiles,CalcMethod,para,gamma);  
%  
%[planC] = dosecalcTOGA(planC,doseindex,Profiles,CalcMethod);  
%  
%Gamma Knife dose calculation  
%planC - CERR treatment planning data  
%doseindex - dosegrid or plan number to be calculated  
%Profiles - Shot 3D profiles used in the calculation  
%CalcMethod - Time or Weight Calc Method (Time = 0, Weight = 1)  
%para - If provided, calculation used in GA optimization para is the  
shot  
%information provided by the GA. The collimator size will be represented  
by a binary  
%variable where [0 0]=4, [0 1]=8, [1 0]=14, [1 1]=18. The remaining  
columns will be the same as the shots variable expect the column index  
will be + 1.  
%gamma - the gamma angle will be provided if an optimization is being  
performed  
%  
% Joseph Bowling  
% University of Tennessee  
% 11-17-2007 - Rewritten 10/7/2010 and 3/3/2011  
%  
%The variable shots is used in the function. The format for the  
variable  
%is:  
% Column1 - Collimator Size  
% Column2 - x  
% Column3 - y  
% Column4 - z  
% Column5 - weight  
% Column6 - gamma  
% Column7 - time  
%  
% Column8 - Collimator factor, added during execution  
% Column9 - Skull attenuation factor, added during execution  
  
%Setup problem  
indexS = planC{end};  
  
%Get shot information, if performing optimization will convert to  
dosecalc  
%shot format  
if nargin == 6  
    %Convert para to shots format for weight calculation  
    para = reshape(para,6,length(para)/6)';  
    numshots = size(para);  
    shots = zeros(numshots(1),6);  
    for i = 1:numshots(1)  
        if isequal(para(i,1:2),[0 0])  
            unbincol(i) = 4;  
        elseif isequal(para(i,1:2),[0 1])  
            unbincol(i) = 8;  
        end  
    end  
end
```

```

elseif isequal(para(i,1:2),[1 0])
    unbincol(i) = 14;
elseif isequal(para(i,1:2),[1 1])
    unbincol(i) = 18;
else
    error
end
end
shots(:,1) = unbincol';
shots(:,2) = para(:,3);
shots(:,3) = para(:,4);
shots(:,4) = para(:,5);
shots(:,5) = para(:,6);
shots(:,6) = gamma;
elseif nargin == 5
    error('Optimization attempted without providing the Gamma angle to
the dose calc')
elseif nargin == 4
    for j = 1:length(planC{indexS.beamGeometry}(doseindex).shots);
        %shots(j,:) =
        [planC{indexS.beamGeometry}(doseindex).shots(1,j).collimator_helmet
planC{indexS.beamGeometry}(doseindex).shots(1,j).targetpointTOGA'
planC{indexS.beamGeometry}(doseindex).shots(1,j).weight
planC{indexS.beamGeometry}(doseindex).shots(1,j).gammaTOGA
planC{indexS.beamGeometry}(doseindex).shots(1,j).time];
        shots(j,:) =
        [planC{indexS.beamGeometry}(doseindex).shots(1,j).collimator_helmet
planC{indexS.beamGeometry}(doseindex).shots(1,j).x
planC{indexS.beamGeometry}(doseindex).shots(1,j).y
planC{indexS.beamGeometry}(doseindex).shots(1,j).z
planC{indexS.beamGeometry}(doseindex).shots(1,j).weight
planC{indexS.beamGeometry}(doseindex).shots(1,j).gammaTOGA
planC{indexS.beamGeometry}(doseindex).shots(1,j).time];
    end
end

%Get doseindex information
calcmatrix =
planC{indexS.beamGeometry}(doseindex).shots(1,1).fractionGroupID;

%Get the dose rate if Time calc method
if CalcMethod == 0
    dr = planC{indexS.dose}(doseindex).doserate;
end

%Discretize shots - Round dose to nearest 0.05
shots(:,1) = round(shots(:,1));
shots(:,[2 3 4]) = round(shots(:,[2 3 4])*20)/20;
shots(:,6) = round(shots(:,6));

%Setup the Dose Calculation
numshots = size(shots);
dcsize = size(planC{indexS.dose}(1,doseindex).doseArray);
dose = zeros(size(planC{indexS.dose}(1,doseindex).doseArray), 'single');

%Add collimator factor to shots variable if Time calc method

```

```

if CalcMethod == 0
    for i = 1:numshots(1)
        switch shots(i,1)
            case 4
                shots(i,8) = 0.870;
            case 8
                shots(i,8) = 0.956;
            case 14
                shots(i,8) = 0.984;
            case 18
                shots(i,8) = 1.000;
        end
    end
end

%Add the attenuation factor to shots variable if Time calc method
if CalcMethod == 0
    r = planC{1,2}.skull.radii;
    %r = 250 - r;
    phi = 90-planC{1,2}.skull.rowAngles;
    phi = repmat(phi,1,8);
    phi = degtorad(phi);
    theta = planC{1,2}.skull.colAngles;
    theta = theta';
    theta = repmat(theta,5,1);
    theta = degtorad(theta);

    [sklx,skly,sklz] = sph2cart(theta,phi,r);
    skl(:, :, 1) = (sklx + 100)/10;
    skl(:, :, 2) = (skly + 100)/10;
    skl(:, :, 3) = (100 - sklz)/10;

    DepthPoints = [reshape(skl(1,8,:),1,3);
    reshape(skl(2:4, :, :),24,3)];
    u = -0.0063; %Linear attenuation coefficient

    %Calculate the depth and add the attenuation factor to the shots
    variable
    for i = 1:numshots(1)
        Depth = mean(sum((DepthPoints -
        repmat(shots(i,1:3),length(DepthPoints),1)).^2,2).^0.5);
        shots(i,9) = exp(u*Depth*10) / exp(u*80);
        %i
        %exp(u*Depth*10) / exp(u*80)
        %2.11 * exp(u*Depth*10) / exp(u*80)
    end
end

%Calculate the dose
for i = 1:numshots(1)
    if shots(i,5) <= 0.04
    else
        rshots = shots(i,2:4);
        scx = find(abs(planC{indexS.dose}(1,doseindex).align.x-rshots(1))
<= 0.04);

```

```

    scy = find(abs(planC{indexS.dose}(1,doseindex).align.y-rshots(2))
<= 0.04);
    scz = find(abs(planC{indexS.dose}(1,doseindex).align.z-rshots(3))
<= 0.04);
    if length(scx) > 1
        scx = scx(1);
    end
    if length(scy) > 1
        scy = scy(1);
    end
    if length(scz) > 1
        scz = scz(1);
    end

    if isempty(scx)
        if rshots(1) > planC{indexS.dose}(1,doseindex).align.x(end)
            extx = (rshots(1) -
planC{indexS.dose}(1,doseindex).align.x(end))/0.05;
            scx = length(planC{indexS.dose}(1,doseindex).align.x) +
extx;
            scx = single(scx);
        elseif rshots(1) < planC{indexS.dose}(1,doseindex).align.x(1)
            extx = (planC{indexS.dose}(1,doseindex).align.x(1) -
rshots(1))/0.05;
            scx = 1 - extx;
            scx = single(scx);
        end
    end
    if isempty(scy)
        if rshots(2) > planC{indexS.dose}(1,doseindex).align.y(end)
            exty = (rshots(2) -
planC{indexS.dose}(1,doseindex).align.y(end))/0.05;
            scy = length(planC{indexS.dose}(1,doseindex).align.y) +
exty;
            scy = single(scy);
        elseif rshots(2) < planC{indexS.dose}(1,doseindex).align.y(1)
            exty = (planC{indexS.dose}(1,doseindex).align.y(1) -
rshots(2))/0.05;
            scy = 1 - exty;
            scy = single(scy);
        end
    end
    if isempty(scz)
        if rshots(3) > planC{indexS.dose}(1,doseindex).align.z(end)
            extz = (rshots(3) -
planC{indexS.dose}(1,doseindex).align.z(end))/0.05;
            scz = length(planC{indexS.dose}(1,doseindex).align.z) +
extz;
            scz = single(scz);
        elseif rshots(3) < planC{indexS.dose}(1,doseindex).align.z(1)
            extz = (planC{indexS.dose}(1,doseindex).align.z(1) -
rshots(3))/0.05;
            scz = 1 - extz;
            scz = single(scz);
        end
    end
end
end

```

```

%Select elements from dose calculation matrix
switch shots(i,1)
    case 4
        xedge = [scx-30 scx+30];
        yedge = [scy-30 scy+30];
        zedge = [scz-30 scz+30];

        pxedge = [1 61];
        pyedge = [1 61];
        pzedge = [1 61];
    case 8
        xedge = [scx-60 scx+60];
        yedge = [scy-60 scy+60];
        zedge = [scz-60 scz+60];

        pxedge = [1 121];
        pyedge = [1 121];
        pzedge = [1 121];
    case 14
        xedge = [scx-75 scx+75];
        yedge = [scy-75 scy+75];
        zedge = [scz-75 scz+75];

        pxedge = [1 151];
        pyedge = [1 151];
        pzedge = [1 151];
    case 18
        xedge = [scx-75 scx+75];
        yedge = [scy-75 scy+75];
        zedge = [scz-75 scz+75];

        pxedge = [1 151];
        pyedge = [1 151];
        pzedge = [1 151];
end

if xedge(1) <= 0
    diff = 1 - xedge(1);
    pxedge(1) = pxedge(1) + diff;
    xedge(1) = 1;
end
if yedge(1) <= 0
    diff = 1 - yedge(1);
    pyedge(1) = pyedge(1) + diff;
    yedge(1) = 1;
end
if zedge(1) <= 0
    diff = 1 - zedge(1);
    pzedge(1) = pzedge(1) + diff;
    zedge(1) = 1;
end
if xedge(2) > dcs(2)
    pxedge(2) = pxedge(2) - (xedge(2) - dcs(2));
    xedge(2) = dcs(2);
end

```

```

    if yedge(2) > dcsz(1)
        pyedge(2) = pyedge(2) - (yedge(2) - dcsz(1));
        yedge(2) = dcsz(1);
    end
    if zedge(2) > dcsz(3)
        pzedge(2) = pzedge(2) - (zedge(2) - dcsz(3));
        zedge(2) = dcsz(3);
    end

    if CalcMethod == 1

        evalstring =
        ['dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) =
dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) +
shots(i,5) *
Profiles.H',num2str(shots(i,1)),'.G',num2str(shots(i,6)),'(pyedge(1):py
edge(2),pxedge(1):pxedge(2),pzedge(1):pzedge(2));'];
        eval(evalstring);

    elseif CalcMethod == 0

        evalstring =
        ['dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) =
dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) + dr *
shots(i,7) * shots(i,8) * shots(i,9) * 0.01 *
Profiles.H',num2str(shots(i,1)),'.G',num2str(shots(i,6)),'(pyedge(1):py
edge(2),pxedge(1):pxedge(2),pzedge(1):pzedge(2));'];
        eval(evalstring);

    end
end

end

%If CalcMethod Weight, Normalize to the maximum dose
if CalcMethod == 1
    dose = planC{indexS.dose}(1,doseindex).Rx * (dose /
max(max(max(dose))));
end

%Write calculated data to planC variable
planC{indexS.dose}(1,doseindex).doseArray = dose;

```


ExtractROIStats

```
function [CurROIData] = ExtractROIStats(structNum,planC);
%[CurROIData] = ExtractROIStats(structNum);
%
%Extracts the ROI parameters to be compared to the database
%
%Joseph Bowling University of TN 2011-09-16

%Initialize the function
indexS = planC{end};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Calculate the comparison parameters for the selected structure
output = structureStats(structNum, planC);

%Get the strucute volume
TV = output.vol;

%Get the largest dimension in X, Y, Z of the structure
LargeDim = output.max - output.min;

%Get the contour points
numslices = length(planC{indexS.structures}(1,structNum).contour);
structPoints = [];
for i = 1:numslices;
    if
        ~isempty(planC{indexS.structures}(1,structNum(1)).contour(i).segments)
            %if
            ~isempty(planC{indexS.structures}(1,structNum(1)).contour(1,i).segments
                .points)
                temp =
                planC{indexS.structures}(1,structNum).contour(i).segments.points;
                %temp =
                planC{indexS.structures}(1,structNum).contour(1,i).segments.points;
                tempsize = size(temp,1);
                strPsize = size(structPoints,1);
                structPoints(strPsize+1:strPsize+tempsize,:) = [temp
                repmat(i,tempsize,1)];
            end
        end
    if ~isempty(planC{indexS.scan}(1,1).transM)
        structPoints(:,3) = structPoints(:,3) +
        planC{indexS.scan}(1,1).transM(3,4);
    else
        structPoints(:,3) = structPoints(:,3);
    end

%Get the center of mass of the structure
%COM = output.COM;
COM = mean(structPoints(:,[1 2 3]));

%Get the raster segments for the structure
[structRasterSegments] = getRasterSegments(structNum, planC);
```

```

%Calculate the spread of the structure from the center of mass
structDistCOM = (sum((structPoints(:,[1 2 3]) -
repmat(COM,size(structPoints,1),1)).^2,2)).^(1/2);
structDistMeanCOM = mean(structDistCOM);
structDistVarCOM = var(structDistCOM);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%PCA analysis
%Perform the PCA
[coeff,score,roots,t2] = princomp(structPoints(:,1:3));
pctExplained = roots' ./ sum(roots);

PC = 2.*sqrt(2.*roots);

%Plot the data with the PCA
figure(10);plot3(structPoints(:,1),structPoints(:,2),structPoints(:,3),
'bo');
grid on;

%Add PCA arrow to graph
LinePoint(1,:) = sqrt(2.*roots(1))*coeff(:,1)';
LinePoint(2,:) = sqrt(2.*roots(2))*coeff(:,2)';
LinePoint(3,:) = sqrt(2.*roots(3))*coeff(:,3)';
LinePoint(1,:) = LinePoint(1,:) + COM;
LinePoint(2,:) = LinePoint(2,:) + COM;
LinePoint(3,:) = LinePoint(3,:) + COM;
line([COM(1) LinePoint(1,1)], [COM(2) LinePoint(1,2)], [COM(3)
LinePoint(1,3)], 'Color', 'r', 'LineWidth', 5);
line([COM(1) LinePoint(2,1)], [COM(2) LinePoint(2,2)], [COM(3)
LinePoint(2,3)], 'Color', 'r', 'LineWidth', 5);
line([COM(1) LinePoint(3,1)], [COM(2) LinePoint(3,2)], [COM(3)
LinePoint(3,3)], 'Color', 'r', 'LineWidth', 5)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Create the CurROIData Variable
CurROIData(1,1) = TV;
CurROIData(1,[2 3 4]) = LargeDim;
CurROIData(1,5) = structDistMeanCOM;
CurROIData(1,6) = structDistVarCOM;
CurROIData(1,[7 8 9]) = PC;

```

ExtractSimPlan

```
function [SelectedShots] = ExtractSimPlan(SelectedPlans);
%[SelectedShots] = ExtractSimPlan(SelectedPlans);
%
%Extracts the shots from the DB that corresponds to the previously
selected
%similar plans.
%
%Joseph Bowling University of TN 2011-09-16

%Load the database
str = computer;
if strcmp(str, 'PCWIN')
    load('C:\TOGA\CERR 3.3 Beta1\UserDefinedFunctions\TOGA\DB.mat');
elseif strcmp(str, 'MACI64')
    load('/TOGA/CERR 3.3 Beta1/UserDefinedFunctions/TOGA/DB.mat');
end

%Find the indices of the individual shots that correspond to the
selected
%plans
count = 1;
SelectedShots = [];
for i = 1:length(SelectedPlans)
    index = find(DB.Shots(:,2) == SelectedPlans(i));
    ExtractShots = DB.Shots(index,3:8);
    %numSelShot = size(ExtractShots,1);
    %SelectedShots(numSelShot+1:numSelShot+length(index),2:7) =
ExtractShots;
    %SelectedShots(numSelShot+1:numSelShot+length(index),1) =
repmat(i,size(SelectedShots,1),1);

    SelectedShots(count:count+length(index)-1,2:7) = ExtractShots;
    SelectedShots(count:count+length(index)-1,1) =
repmat(i,size(ExtractShots,1),1);
    count = size(SelectedShots,1) + 1;
end
```

FTPGamma2Cerr

```
function FTPGamma2Cerr(directory);
%Funtion that FTPs treatment planning data from GammaPlan treatment
%planning system to local computer.
%
%   FTPGamma2Cerr(directory);
%   directory - local directory to save treatment planning data
%
%Joseph Bowling
%University of Tennessee
%12-10-2007

%Select directory to save GammaPlan files on local box
n = nargin;
if n == 0
    directory = uigetdir;
end

%Original Directory
OrgDir = pwd;

%Create connection to FTP server
GammaPlan = ftp('172.30.103.232','lgp','elekta');

%Change to GammaPlan database directory
cd(GammaPlan, '/opt/elekta/lgp/dbs');

%Get list of available patients
patients_dir = dir(GammaPlan);
for i = 1:length(patients_dir)
    sara = [num2str(i) ': ' patients_dir(i).name];
    disp(sara);
end
selection = input('Enter the number(s) of the plans to ftp ([x:y] or [x
y z] format for multiple plans: ');
selection = selection';
%close(GammaPlan);
%clear GammaPlan;

%Get information for change of directory
for i = 1:length(selection);
    %GammaPlan = ftp('172.30.103.232','lgp','elekta');
    selDir = patients_dir(selection(i)).name;
    %Change to and make the appropriate directories
    ftpDir = ['/opt/elekta/lgp/dbs/' selDir];
    cd(GammaPlan, ftpDir);
    cd(directory);
    mkdir(selDir);
    localDir = [directory '\\' selDir];
    cd(localDir);

    %FTP Images
    dbFolders = dir(GammaPlan);
    for j = 1:length(dbFolders);
```

```

        if strcmp('MR',dbFolders(j).name);
            %Create Image Directory on Local Box
            remain = dbFolders(j).name;
            while ~isempty(remain)
                [token,remain] = strtok(remain, ':');
            end
            [token,remain] = strtok(token, '.');
            ImageDir = ['MR-' token];
            mkdir(ImageDir);
            %FTP Files from GammaPlan to Image Directory on Local Box
            ftpImageDir = [ftpDir '/' dbFolders(j).name];
            cd(GammaPlan,ftpImageDir);
            localImageDir = [localDir '\\' ImageDir];
            cd(localImageDir);
            mget(GammaPlan, '*.*');
            cd(localDir);
            cd(GammaPlan,ftpDir);
        end
    end
    close(GammaPlan);
    %clear GammaPlan;

    %Recreate connection to FTP server
    GammaPlan2 = ftp('172.30.103.232','lgp','elekta');
    %Change to GammaPlan database directory
    cd(GammaPlan2,ftpDir);

    %FTP everything else except snapshots
    dbFolders2 = dir(GammaPlan2);
    mget(GammaPlan2,'Fusions');
    for k = 1:length(dbFolders)
        if strcmpi(dbFolders(k).name, 'Logs')
            mget(GammaPlan2,'Logs');
        end
    end
    mget(GammaPlan2,'Orientations');
    mget(GammaPlan2,'Plugs');
    mget(GammaPlan2,'Shots');
    %mget(GammaPlan2,'Snapshots');
    mget(GammaPlan2,'Sortings');
    mget(GammaPlan2,'SRdata');
    mget(GammaPlan2,'Studyregs');
    mget(GammaPlan2,'Studys');
    mget(GammaPlan2,'Thresholds');
    mget(GammaPlan2,'Volumes');
    mget(GammaPlan2,'Workspaces');

    close(GammaPlan2);
    clear GammaPlan2;

    cd(directory);
end

```

GACreationUniformTOGA

```
function Population =
gacreationuniformTOGA(GenomeLength,FitnessFcn,options)
%GACREATIONUNIFORMTOGA Creates the initial population for genetic
algorithm.
%
%   The initial population will be a combination of Bit String
(Logical)
%   and Double type data.  The first two columns of the data will be a
bit
%   string representing the Gaussian Sigma or for the Gamma Knife
%   optimization the collimator size.  Only four sigmas or collimator
sizes
%   will be available.  The remaining columns will be the dimensional
%   coordinates and the weight of the Gaussian Curve or the Collimator
%   shot.
%
%   POP = GACREATIONUNIFORM4DoubleBitData(NVARS,FITNESSFCN,OPTIONS)
Creates the
%   initial population that GA will then evolve into a solution.
%
%   Example:
%       options = gaoptimset('CreationFcn',@gacreationuniform);

%   Copyright 2003-2009 The MathWorks, Inc.
%   $Revision: 1.1.6.3 $   $Date: 2009/11/05 16:59:24 $
%   Revised by Joseph Bowling University of Tennessee 2/21/2011

fprintf('Creation Function has run.  The size of the initial pop is
%d.\n\n', size(options.InitialPopulation,1));

size(options.InitialPopulation)

%Problem setup
totalPopulation = sum(options.PopulationSize);

initPopProvided = size(options.InitialPopulation,1)
%initPopProvided = 0;
individualsToCreate = totalPopulation - initPopProvided
numshots = GenomeLength / 6;

% Initialize Population to be created
Population = zeros(totalPopulation,GenomeLength);
% Use initial population provided already
if initPopProvided > 0
    Population(1:initPopProvided,:) = options.InitialPopulation;
end

% Create remaining double initial population
range = options.PopInitRange;
lowerBound = range(1,:);
span = range(2,:) - lowerBound;
Population(initPopProvided+1:end,:) =
repmat(lowerBound,individualsToCreate,1) + ...
```

```

repmat(span, individualsToCreate, 1) .*
rand(individualsToCreate, GenomeLength);

% Create remaining bit initial population
BitPopulation = ones(totalPopulation, GenomeLength);
% Use initial population provided already
if initPopProvided > 0
    BitPopulation(1:initPopProvided,:) = options.InitialPopulation;
end
% Create remaining population
BitPopulation(initPopProvided+1:end,:) = double(0.5 >
rand(individualsToCreate, GenomeLength));

%Combine created Double and Bit initial population
for i = 1:numshots
    count = [1+(i-1)*6 2+(i-1)*6];
    Population(:,count) = BitPopulation(:,count);
end

%Check the created population
if all(isnan(Population))
    error('globaloptim:gacreationuniform:populationIsNaN', ['Initial
population contains NaN;', 'OPTIONS.PopInitRange is possibly too
big.']);
elseif all(isinf(Population))
    error('globaloptim:gacreationuniform:populationIsInf', ['Initial
population contains Inf;', 'OPTIONS.PopInitRange is possibly too
big.']);
end

```

GAInitialPopulation

```
function [planC] = GAInitialPopulation(planC, structNum, doseindex,
gamma, method);
%[planC] = GAInitialPopulation(planC, structNum, method);
%
%GAInitialPopulation provides an initial population for the Genetic
%Algorithm optimization of the planC dose matrix based on the selected
%structure and the specified method
%
%planC - CERR structure
%structNum - Structure index to provide initial plan
%doseindex - Dose matrix index of the plan to be optimized
%gamma - gamma angle of shots for the initial population
%method - Matrix specifying method and information to determine initial
plan
% method(1,1) - Use initial population contained in planC variable
(0/1)
% method(2,1) - Plan index to be used for initial population if
% method(1,1) = 1
% method(1,2) - Use StructMat of 0/1 to create initial plan via
geometric
% optimization method
% method(2,2) - Packing parameter for geometric optimization (0.2-
0.8)
% method(3,2) - Number of shots for geometric optimization
% method(1,7) - Extract prior plans from database for optimization
% method(2,7) - Number of plans to extract
%
%Joseph Bowling University of TN 5/3/2011
%
%method = [0 1 1 1 1 1 1;0 0.7 0.8 0.6 0.8 0.5 5;0 13 13 13 13 13 0]

%Setup the problem
indexS = planC{end};
warning off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Provide an intial population from an existing plan
countX0 = 1;
if method(1)
    for i = 1:length(planC{indexS.beamGeometry})
        shots = planC{indexS.beamGeometry}(method(2,1),1).shots;
    end
    for i = 1:length(shots)
        X0Exist(i,3) = shots(method(2,1),i).x;
        X0Exist(i,4) = shots(method(2,1),i).y;
        X0Exist(i,5) = shots(method(2,1),i).z;
        X0Exist(i,6) = shots(method(2,1),i).weight;
        X0Exist(i,1) = shots(method(2,1),i).collimator_helmet;
    end
    for i = 1:length(shots)
        if isequal(X0Exist(i,1),4)
            X0Exist(i,1:2) = [0 0];
        elseif isequal(X0Exist(i,1),8)
```



```

        X0Exist(i,1:2) = [0 1];
    elseif isequal(X0Exist(i,1),14)
        X0Exist(i,1:2) = [1 0];
    elseif isequal(X0Exist(i,1),18)
        X0Exist(i,1:2) = [1 1];
    else
        error
    end
end
%Reshape X0Exist variable
X0Existsize = size(X0Exist);
X0Exist = reshape(X0Exist',1,X0Existsize(1)*X0Existsize(2));
%Add E0Exist to the X0 variable
X0{countX0,:} = X0Exist;
countX0 = countX0 + 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Create a matrix of 0/1 that represents the volume to be optimized.
[x1V, y1V, z1V] = getVolumePoints(structNum, doseindex, planC);
%x1V,y1V,z1V are XYZ coordinates of points that make up the structure
VolPoints = [x1V' y1V' z1V'];
%Discretize VolPoints to the nearest 0.05
VolPoints = round(VolPoints*20)/20;
VolPoints = unique(VolPoints, 'rows');
MatSize = size(planC{indexS.dose}(doseindex).doseArray);
StructMat = zeros(MatSize(1),MatSize(2),MatSize(3),'single');
MatIndX = planC{indexS.dose}(doseindex).align.x;
MatIndY = planC{indexS.dose}(doseindex).align.y;
MatIndZ = planC{indexS.dose}(doseindex).align.z;
%Find the volume point index and place a 1 in the volume matrix
for i = 1:length(VolPoints)
    vpx = find(abs(MatIndX-VolPoints(i,1)) <= 0.04);
    vpy = find(abs(MatIndY-VolPoints(i,2)) <= 0.04);
    vpz = find(abs(MatIndZ-VolPoints(i,3)) <= 0.04);
    StructMat(vpx,vpy,vpz) = 1;
end
%Smooth the 3D StructMat to fill in holes
StructMat = smooth3(StructMat,'gaussian',[1 1 3],0.65);
StructMat = ceil(StructMat);
%Rotate the structure matrix
for i = 1:size(StructMat,3)
    a = rot90(StructMat(:, :, i));
    StructMatRot(:, :, i) = flipud(a);
end
StructMat = StructMatRot;
%Plot the matrix of the structure
%for i = 1:size(StructMat,3)
%    a = StructMat(:, :, i);
%    figure(10);imagesc(a);
%    %b = StructMat(:, :, i);
%    %figure(11);imagesc(b);
%    pause(0.1)
%end

```

```

%The prior code created a 3D matrix with 1s representing each point of
the
%structre in the matrix. The variables MatIndX,Y,Z represent the
%coordinates of the matrix points in the CERR coordinate system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
NumPlanSel = 0;
%Kernel Based Prior Plan Selection
if method(1,7) == 1
    %Extract the ROI parameters from the ROI to be optimized for
comparison to
    %the database
    [CurROIData] = ExtractROIStats(structNum,planC)

    %Compare the ROI parameters to the database and extract the index
of the
    %similar plans
    NumPlanSel = method(2,7); %Number of plans to extract from the
database
    [SelectedPlans,Similar] = Compared2DB(CurROIData,NumPlanSel)

    %Extract the shots for the selected plans from the database
    [SelectedShots] = ExtractSimPlan(SelectedPlans);

    %Transform the selected plans to the current ROI to be optimized
    for j = 1:NumPlanSel
        index = find(SelectedShots(:,1) == j);
        SelectShots = SelectedShots(index,2:6);

        %Transform the prior created treatment plans that were selected
by the
        %kernel selection to the current optimization volume
        %Calculate the center of the optimization volume
        CenterContour = [mean(x1V) mean(y1V) mean(z1V)];

        %Transform the Selected plans to the current structure to be
optimized
        tic;
        [TransformShots] = TransformPlan(planC, StructMat, structNum,
doseindex, gamma, method, MatIndX, MatIndY, MatIndZ, SelectShots,
CenterContour);
        clear X0PriorPlan;
        X0PriorPlan(:,[3:6]) = TransformShots(:,2:5);
        for i = 1:size(TransformShots,1)
            if isequal(TransformShots(i,1),4)
                X0PriorPlan(i,1:2) = [0 0];
            elseif isequal(TransformShots(i,1),8)
                X0PriorPlan(i,1:2) = [0 1];
            elseif isequal(TransformShots(i,1),14)
                X0PriorPlan(i,1:2) = [1 0];
            elseif isequal(TransformShots(i,1),18)
                X0PriorPlan(i,1:2) = [1 1];
            else

```

```

        error
    end
    end
    X0PriorPlan =
reshape(X0PriorPlan',1,size(X0PriorPlan,1)*size(X0PriorPlan,2));
    X0{countX0,:} = X0PriorPlan;
    countX0 = countX0 + 1;
    tElapsed=toc;
    timeprint(j) = tElapsed;
    Message = ['Transformed the prior selected plan to the current
optimization structure in ', num2str(tElapsed), ' sec',,];
    h = msgbox(Message,'Plan Transform');
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Create an initial population via a geometric optimization method
utilizing
%the matrix of 0/1 representing the structure (Attempt 1)
if method(1,2) == 1
    tic;
    [X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method(:,2), MatIndX, MatIndY, MatIndZ);
    %Add E0Exist to the X0 variable
    X0{countX0,:} = X0Geo;
    countX0 = countX0 + 1;
    clear X0Geo;
    tElapsed=toc;
    timeprint(NumPlanSel+1) = tElapsed;
    Message = ['First optimization completed in ', num2str(tElapsed), '
sec',,];
    h = msgbox(Message,'Geometric Optimization');
end

%Create an initial population via a geometric optimization method
utilizing
%the matrix of 0/1 representing the structure (Attempt 2)
if method(1,3) == 1
    tic;
    [X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method(:,3), MatIndX, MatIndY, MatIndZ);
    %Add E0Exist to the X0 variable
    X0{countX0,:} = X0Geo;
    countX0 = countX0 + 1;
    clear X0Geo;
    tElapsed=toc;
    timeprint(NumPlanSel+2) = tElapsed;
    Message = ['Second optimization completed in ', num2str(tElapsed),
' sec',,];
    h = msgbox(Message,'Geometric Optimization');
end

```

```

%Create an initial population via a geometric optimization method
utilizing
%the matrix of 0/1 representing the structure (Attempt 3)
if method(1,4) == 1
    tic;
    [X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method(:,4), MatIndX, MatIndY, MatIndZ);
    %Add E0Exist to the X0 variable
    X0{countX0,:} = X0Geo;
    countX0 = countX0 + 1;
    clear X0Geo;
    tElapsed=toc;
    timeprint(NumPlanSel+3) = tElapsed;
    Message = ['Third optimization completed in ', num2str(tElapsed), '
sec',];
    h = msgbox(Message,'Geometric Optimization');
end

%Create an initial population via a geometric optimization method
utilizing
%the matrix of 0/1 representing the structure (Attempt 4)
if method(1,5) == 1
    tic;
    [X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method(:,5), MatIndX, MatIndY, MatIndZ);
    %Add E0Exist to the X0 variable
    X0{countX0,:} = X0Geo;
    countX0 = countX0 + 1;
    clear X0Geo;
    tElapsed=toc;
    timeprint(NumPlanSel+4) = tElapsed;
    Message = ['Forth optimization completed in ', num2str(tElapsed), '
sec',];
    h = msgbox(Message,'Geometric Optimization');
end

%Create an initial population via a geometric optimization method
utilizing
%the matrix of 0/1 representing the structure (Attempt 5)
if method(1,6) == 1
    tic;
    [X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method(:,6), MatIndX, MatIndY, MatIndZ);
    %Add E0Exist to the X0 variable
    X0{countX0,:} = X0Geo;
    countX0 = countX0 + 1;
    clear X0Geo;
    tElapsed=toc;
    timeprint(NumPlanSel+5) = tElapsed;
    Message = ['Fifth optimization completed in ', num2str(tElapsed), '
sec',];
    h = msgbox(Message,'Geometric Optimization');
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%Alter X0 so that each plan is the same length. This involves adding
dummy
%4 mm shots with zero weight. These dummy four shots will be added
%homogeneously throughout the ROI to be optimized
VolSpread = [min(VolPoints); max(VolPoints)];
for i = 1:length(X0)
    MaxNumShots(i) = length(X0{i}(:));
end
MaxNumShots = max(MaxNumShots);
for i = 1:length(X0)
    numshots2add(i) = (MaxNumShots - length(X0{i}(:))) / 6;
end
for i = 1:length(X0)
    if numshots2add(i) > 0
        ShotsAdd = zeros(numshots2add(i), 6);
        ShotsAdd(:, [3]) = VolSpread(1,1) + (VolSpread(2,1)-
VolSpread(1,1)).*rand(numshots2add(i),1);
        ShotsAdd(:, [4]) = VolSpread(1,2) + (VolSpread(2,2)-
VolSpread(1,2)).*rand(numshots2add(i),1);
        ShotsAdd(:, [5]) = VolSpread(1,3) + (VolSpread(2,3)-
VolSpread(1,3)).*rand(numshots2add(i),1);
        ShotsAdd =
        reshape(ShotsAdd', 1, size(ShotsAdd,1)*size(ShotsAdd,2));
        X0{i} = [X0{i} ShotsAdd];
    end
end

%Assign created initial population to the planC variable in the comment
%section
planC{indexS.comment}.GAOptInitPopulation = X0;

%Update planC with the new dose matrix position
assignin('base', 'planC', planC);

timeprint'
sum(timeprint)
mean(timeprint)

```

GAInitialPopulationGUI

```
function varargout = GAInitialPopulationGUI(varargin)
% GAINITIALPOPULATIONGUI MATLAB code for GAInitialPopulationGUI.fig
%     GAINITIALPOPULATIONGUI, by itself, creates a new
GAINITIALPOPULATIONGUI or raises the existing
%     singleton*.
%
%     H = GAINITIALPOPULATIONGUI returns the handle to a new
GAINITIALPOPULATIONGUI or the handle to
%     the existing singleton*.
%
%     GAINITIALPOPULATIONGUI('CALLBACK',hObject,eventData,handles,...)
calls the local
%     function named CALLBACK in GAINITIALPOPULATIONGUI.M with the
given input arguments.
%
%     GAINITIALPOPULATIONGUI('Property','Value',...) creates a new
GAINITIALPOPULATIONGUI or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before GAInitialPopulationGUI_OpeningFcn gets
called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to
GAInitialPopulationGUI_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
GAInitialPopulationGUI

% Last Modified by GUIDE v2.5 11-Nov-2011 15:36:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @GAInitialPopulationGUI_OpeningFcn, ...
                  'gui_OutputFcn',   @GAInitialPopulationGUI_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```

else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GAInitialPopulationGUI is made visible.
function GAInitialPopulationGUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to GAInitialPopulationGUI (see
VARARGIN)

% Choose default command line output for GAInitialPopulationGUI
handles.output = hObject;

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
end

%Populate objects in the GUI
setGUIpopupmenuPlan(handles);
setGUIpopupmenuROI(handles);
setGUILabelSlideValue(handles);

set(handles.popupmenuGamma,'String',{'72' '90' '110' '125'});
set(handles.popupmenuGamma,'Value',2);

set(handles.popupmenu6,'String',{'4' '8' '14' '18'});
set(handles.popupmenu6,'Value',4);

set(handles.popupmenu7,'String',{'4' '8' '14' '18'});
set(handles.popupmenu7,'Value',4);

set(handles.popupmenu8,'String',{'4' '8' '14' '18'});
set(handles.popupmenu8,'Value',4);

set(handles.popupmenu9,'String',{'4' '8' '14' '18'});
set(handles.popupmenu9,'Value',4);

set(handles.popupmenu10,'String',{'4' '8' '14' '18'});
set(handles.popupmenu10,'Value',4);

```

```

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GAInitialPopulationGUI wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = GAInitialPopulationGUI_OutputFcn(hObject,
eventdata, handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function setGUIpopupmenuPlan(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)
    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
end
set(handles.popupmenuPlan, 'String', plannames);
set(handles.popupmenu11, 'String', plannames);

function setGUIpopupmenuROI(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the ROIs names
ROIsize = size(handles.planC{handles.indexS.structures});
for i = 1:ROIsize(2)
    ROInames{i,1} =
handles.planC{handles.indexS.structures}(1,i).structureName;
end
set(handles.popupmenuStructure, 'String', ROInames);

% --- Executes on selection change in popupmenuPlan.
function popupmenuPlan_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB

```



```

% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuPlan contents as cell array
%      contents{get(hObject,'Value')} returns selected item from
popupmenuPlan

% --- Executes during object creation, after setting all properties.
function popupmenuPlan_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuStructure.
function popupmenuStructure_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuStructure (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuStructure contents as cell array
%      contents{get(hObject,'Value')} returns selected item from
popupmenuStructure

% --- Executes during object creation, after setting all properties.
function popupmenuStructure_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuStructure (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuGamma.
function popupmenuGamma_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuGamma (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuGamma contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuGamma

% --- Executes during object creation, after setting all properties.
function popupmenuGamma_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuGamma (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Get the selections
structNum = get(handles.popupmenuStructure,'Value');
doseindex = get(handles.popupmenuPlan,'Value');
gammaSel = get(handles.popupmenuGamma,'Value');
switch gammaSel
    case 1
        gamma = 72;
    case 2
        gamma = 90;
    case 3
        gamma = 110;
    case 4
        gamma = 125;
end

%Create the method variable
method = zeros(4,7);
checkvalue = get(handles.checkbox6,'Value');
if checkvalue == 1
    method(1,1) = 1;
    method(2,1) = get(handles.popupmenu11,'Value');
end

checkvalue = get(handles.checkbox7,'Value');
if checkvalue == 1
    method(1,7) = 1;

```

```

        numhistplans = get(handles.edit2, 'String');
        method(2,7) = str2num(numhistplans);
end

GeoShotNum = get(handles.edit3, 'String');
GeoShotNum = str2num(GeoShotNum);
checkvalue = get(handles.checkbox1, 'Value');
if checkvalue == 1
    method(1,2) = 1;
    method(2,2) = get(handles.slider1, 'Value');
    method(3,2) = GeoShotNum;
    colSel = get(handles.popupmenu6, 'Value');
    switch colSel
        case 1
            method(4,2) = 4;
        case 2
            method(4,2) = 8;
        case 3
            method(4,2) = 14;
        case 4
            method(4,2) = 18;
    end
end

checkvalue = get(handles.checkbox2, 'Value');
if checkvalue == 1
    method(1,3) = 1;
    method(2,3) = get(handles.slider2, 'Value');
    method(3,3) = GeoShotNum;
    colSel = get(handles.popupmenu7, 'Value');
    switch colSel
        case 1
            method(4,3) = 4;
        case 2
            method(4,3) = 8;
        case 3
            method(4,3) = 14;
        case 4
            method(4,3) = 18;
    end
end

checkvalue = get(handles.checkbox3, 'Value');
if checkvalue == 1
    method(1,4) = 1;
    method(2,4) = get(handles.slider3, 'Value');
    method(3,4) = GeoShotNum;
    colSel = get(handles.popupmenu8, 'Value');
    switch colSel
        case 1
            method(4,4) = 4;
        case 2
            method(4,4) = 8;
        case 3
            method(4,4) = 14;
        case 4

```

```

        method(4,4) = 18;
    end
end

checkvalue = get(handles.checkbox4, 'Value');
if checkvalue == 1
    method(1,5) = 1;
    method(2,5) = get(handles.slider4, 'Value');
    method(3,5) = GeoShotNum;
    colSel = get(handles.popupmenu9, 'Value');
    switch colSel
        case 1
            method(4,5) = 4;
        case 2
            method(4,5) = 8;
        case 3
            method(4,5) = 14;
        case 4
            method(4,5) = 18;
    end
end

checkvalue = get(handles.checkbox5, 'Value');
if checkvalue == 1
    method(1,6) = 1;
    method(2,6) = get(handles.slider5, 'Value');
    method(3,6) = GeoShotNum;
    colSel = get(handles.popupmenu10, 'Value');
    switch colSel
        case 1
            method(4,6) = 4;
        case 2
            method(4,6) = 8;
        case 3
            method(4,6) = 14;
        case 4
            method(4,6) = 18;
    end
end

method

[handles.planC] = GAInitialPopulation(handles.planC, structNum,
doseindex, gamma, method);

%Update planC with the new dose matrix position
assignin('base', 'planC', handles.planC);
guidata(hObject, handles);

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

```

```

% Hint: get(hObject,'Value') returns toggle state of checkbox1

% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2

% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3

% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox4

% --- Executes on button press in checkbox5.
function checkbox5_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox5

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUILabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)

```

```

% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function setGUILabelSlideValue(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

slidervalue = get(handles.slider1,'Value');
slidervalue = num2str(slidervalue);
set(handles.text7,'String',slidervalue);

slidervalue = get(handles.slider2,'Value');
slidervalue = num2str(slidervalue);
set(handles.text8,'String',slidervalue);

slidervalue = get(handles.slider3,'Value');
slidervalue = num2str(slidervalue);
set(handles.text9,'String',slidervalue);

slidervalue = get(handles.slider4,'Value');
slidervalue = num2str(slidervalue);
set(handles.text10,'String',slidervalue);

slidervalue = get(handles.slider5,'Value');
slidervalue = num2str(slidervalue);
set(handles.text11,'String',slidervalue);

% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject      handle to slider2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUILabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to slider2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB

```

```

% handles      empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject      handle to slider3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%          get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUILabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject      handle to slider3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject      handle to slider4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%          get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUILabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject      handle to slider4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

```

```

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject      handle to slider5 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%         get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUIDebugLabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject      handle to slider5 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on selection change in popupmenu6.
function popupmenu6_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu6 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu6
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenu6

% --- Executes during object creation, after setting all properties.
function popupmenu6_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu6 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.

```



```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu7.
function popupmenu7_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu7
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu7

% --- Executes during object creation, after setting all properties.
function popupmenu7_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu8.
function popupmenu8_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu8 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu8
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu8

% --- Executes during object creation, after setting all properties.
function popupmenu8_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu8 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu9.
function popupmenu9_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu9
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenu9

% --- Executes during object creation, after setting all properties.
function popupmenu9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu10.
function popupmenu10_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu10
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenu10

% --- Executes during object creation, after setting all properties.
function popupmenu10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu11.
function popupmenu11_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu11 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu11
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu11

% --- Executes during object creation, after setting all properties.
function popupmenu11_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu11 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox6.
function checkbox6_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox6 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox6

checkvalue = get(handles.checkbox6,'Value');
if checkvalue == 1
    set(handles.text15,'visible','on');
    set(handles.popupmenu11,'visible','on');
else
    set(handles.text15,'visible','off');
    set(handles.popupmenu11,'visible','off');
end

% --- Executes on button press in checkbox7.
function checkbox7_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox7 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox7


function edit2_Callback(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%         str2double(get(hObject,'String')) returns contents of edit2 as
a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit3_Callback(hObject, eventdata, handles)
% hObject      handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%         str2double(get(hObject,'String')) returns contents of edit3 as
a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))

```

```
        set(hObject,'BackgroundColor','white');  
    end
```

GAPlotConformIndexOpt

```
function state = GAPlotConformIndexOpt(options, state, flag, planC,
doseindex, Profiles, CalcMethod, gamma, structNum, RxIsodose,TV);
% GAPlotConformIndexOpt plots the conformity indices of the optimal
plan
%
persistent last_best; % Best score in the previous generation

if(strcmp(flag,'init')) % Set up the plot
    %set(gca);
    title('DVH of Best Individual');
end

%Calculated the DVH for the best individual at the current generation
index = find(state.Score == min(state.Score)); % index of the best
score in the current generation
index = index(1);
para = state.Population(index,:);
reshape(para,6,length(para)/6)';

%Calculate the dose distribution for the best individual
[planC] = dosecalcTOGA(planC,doseindex,Profiles,CalcMethod,para,gamma);

%Calculate the conformity
if size(structNum,1) >= 2;
    if structNum(2,1) ~= 0

[Iso_Cov,Cov_RT0G,CI_RT0G,CI_Paddick,CI_Gradient,CI_Karolinska, RingInde
x] = ConformIndex(planC, structNum, doseindex, RxIsodose, TV);
        else
            [Iso_Cov,Cov_RT0G,CI_RT0G,CI_Paddick,CI_Gradient,CI_Karolinska]
= ConformIndex(planC, structNum, doseindex, RxIsodose, TV);
        end
    else
        [Iso_Cov,Cov_RT0G,CI_RT0G,CI_Paddick,CI_Gradient,CI_Karolinska] =
ConformIndex(planC, structNum, doseindex, RxIsodose, TV);
    end

%Plot the indices
if size(structNum,1) >= 2;
    if structNum(2,1) ~= 0
        bar([-1*Iso_Cov)+1 (-1*Cov_RT0G)+1 CI_RT0G (-1*CI_Paddick)+1
CI_Gradient - 2.5 CI_Karolinska RingIndex]);
        text(7,0,'RingIndex','Rotation',90,'Color','Red');
    else
        bar([-1*Iso_Cov)+1 (-1*Cov_RT0G)+1 CI_RT0G (-1*CI_Paddick)+1
CI_Gradient - 2 CI_Karolinska]);
    end
else
    bar([-1*Iso_Cov)+1 (-1*Cov_RT0G)+1 CI_RT0G (-1*CI_Paddick)+1
CI_Gradient - 2 CI_Karolinska]);
end
xlabel('Conformity Indices');
text(1,0,'% Isodose Coverage','Rotation',90,'Color','Red');
```

```

text(2,0,'RTOG Coverage','Rotation',90,'Color','Red');
text(3,0,'RTOG Conformity','Rotation',90,'Color','Red');
text(4,0,'Paddick Conformity','Rotation',90,'Color','Red');
text(5,0,'Gradient Index','Rotation',90,'Color','Red');
text(6,0,'Karolinsak Index','Rotation',90,'Color','Red');
%
% if size(structNum,1) >= 3;
%     [dosesV, volsV, isError] = getDVH(structNum(3,1), doseindex,
planC);
%     f = find(dosesV>structNum(3,2));
%     if length(f) == 0
%         JMan =0
%     else
%         JMan = sum((dosesV(f)-structNum(3,2)).^2)/length(dosesV(f))
%     end
% else
%     JMan = 0;
% end
%
%fprintf('Next Generation\n\n');

```

GeoGAOptFunc

```
function [Cost] =
GeoGAOptFunc(para, StructMat, MatIndX, MatIndY, MatIndZ, gamma, Profiles, RxIs
odose, Helmet, VolInit);
%
%[Cost] =
GeoGAOptFunc(para, StructMat, MatIndX, MatIndY, MatIndZ, gamma, Profiles, RxIs
odose);
%
%Cost function for the geometric optimization of seeding Gamma Knife
treatment plans with
%Genetic Algorithms.
%
%Written by Joseph Bowling University of Tennessee 5/12/2011

%Calculate the geometric coverage with the shot arrangement provided by
the
%genetic algorithm
shots = [Helmet para 1];
[StructMat] =
StructMatGeometricCalc(StructMat, MatIndX, MatIndY, MatIndZ, shots, gamma, Pro
files, RxIsodose);

%Calculated the portion of the target not covered
Cost = sum(sum(sum(StructMat)))/VolInit;
```


GeometricOptimization

```

function [X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method, MatIndX, MatIndY, MatIndZ);
%
%[X0Geo] = GeometricOptimization(planC, StructMat, structNum,
doseindex, gamma, method);
%
%Geometric optimization of the treatment volume via genetic algorithm
%
%Joseph Bowling University of TN 5/11/2011

%Setup the problem
indexS = planC{end};

%Placed specified number of shots utilizing a geometric optimization
via GA
StructMatGeo = StructMat;
VolInit = sum(sum(sum(StructMatGeo)));
TV = getStructureVol(structNum,planC);
RatioTV = TV / VolInit;
[Profiles] = loadprofilesTOGA(planC,doseindex,gamma);
RxIsodose = method(2);

%Using the volume to determine the helmets to use
%Helmet = [18 8 8 8 8 4 4 4 4 4 4 4 4];
Helmet = zeros(1,method(3)); %method(3) is the number of shots for the
optimization
%
%      40%      50%      60%      70%      80%      90%
HelmetVols = [0.1244 0.0894 0.066 0.0479 0.0318 0.0135; %4
0.7988 0.5885 0.4534 0.3538 0.2625 0.1461; %8
3.9 2.9 2.3 1.9 1.6 1.2; %14
7.7 5.8 4.7 4 3.4 2.7]; %18
HelmetVolIso = [.40 .50 .60 .70 .80 .90];
if RxIsodose <= 0.4
    HelmetVolumes = HelmetVols(:,1);
elseif RxIsodose >= 0.9
    HelmetVolumes = HelmetVols(:,6);
else
    HelInd = find(round(RxIsodose*10)/10==HelmetVolIso);
    HelmetVolumes = HelmetVols(:,HelInd);
end

%Define the optimization function
OptFunc =
@(para) GeoGAOptFunc(para, StructMatGeo, MatIndX, MatIndY, MatIndZ, gamma, Pro
files, RxIsodose, Helmet, VolInit);

%Determine the range of values for the parameters (ie spread of x,y,z
of
%the target volume) for the parameter optimization range.
numslices = length(planC{indexS.structures}(1,structNum).contour);
count = 1;
for i = 1:numslices;
    if ~isempty(planC{indexS.scan}(1,1).transM)

```

```

        if
~isempty(planC{indexS.structures}(1,structNum).contour(1,i).segments.points)
            x(count,:) =
[min(planC{indexS.structures}(1,structNum).contour(1,i).segments.points(:,1))
max(planC{indexS.structures}(1,structNum).contour(1,i).segments.points(:,1))];
            y(count,:) =
[min(planC{indexS.structures}(1,structNum).contour(1,i).segments.points(:,2))
max(planC{indexS.structures}(1,structNum).contour(1,i).segments.points(:,2))];
            z(count,:) =
[min(planC{indexS.structures}(1,structNum).contour(1,i).segments.points(:,3))
max(planC{indexS.structures}(1,structNum).contour(1,i).segments.points(:,3))];
            count = count + 1;
        end
    else
        if
~isempty(planC{indexS.structures}(1,structNum).contour(i).segments)
            x(count,:) =
[min(planC{indexS.structures}(1,structNum).contour(i).segments(1).points(:,1))
max(planC{indexS.structures}(1,structNum).contour(i).segments(1).points(:,1))];
            y(count,:) =
[min(planC{indexS.structures}(1,structNum).contour(i).segments(1).points(:,2))
max(planC{indexS.structures}(1,structNum).contour(i).segments(1).points(:,2))];
            z(count,:) =
[min(planC{indexS.structures}(1,structNum).contour(i).segments(1).points(:,3))
max(planC{indexS.structures}(1,structNum).contour(i).segments(1).points(:,3))];
            count = count + 1;
        end
    end
end

if ~isempty(planC{indexS.scan}(1,1).transM)
    Lower = [0 0 min(x(:,1)) min(y(:,1)) min(z(:,1)) 0.1];
    Upper = [1 1 max(x(:,2)) max(y(:,2)) max(z(:,2)) 1.4];
    Lower(3) = Lower(3) + planC{1,3}(1,1).transM(1,4);
    Upper(3) = Upper(3) + planC{1,3}(1,1).transM(1,4);
    CalcUp4 = planC{1,3}(1,1).transM(2,4) - Lower(4);
    CalcLow4 = planC{1,3}(1,1).transM(2,4) - Upper(4);
    Lower(4) = CalcLow4;
    Upper(4) = CalcUp4;
    Lower(5) = Lower(5) + planC{1,3}(1,1).transM(3,4);
    Upper(5) = Upper(5) + planC{1,3}(1,1).transM(3,4);
    LowUp = [Lower;Upper];
end

```

```

else
    Lower = [0 0 min(x(:,1)) min(y(:,1)) min(z(:,1)) 0.1];
    Upper = [1 1 max(x(:,2)) max(y(:,2)) max(z(:,2)) 1.4];
    LowUp = [Lower;Upper];
end

%Setup the GA optimization
nvars = 3;
PopInitRange_Data = LowUp(:,3:5);
PopulationSize_Data = 7;
Generations_Data = 10;
lb = LowUp(1,3:5);
ub = LowUp(2,3:5);

% Start with the default options
options = gaoptimset;
% Modify options setting
options = gaoptimset(options,'PopInitRange', PopInitRange_Data);
options = gaoptimset(options,'PopulationSize', PopulationSize_Data);
options = gaoptimset(options,'Generations', Generations_Data);
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', { @gaplotbestf @gaplotscores
});
%options = gaoptimset(options,'UseParallel', 'always');

%Using a GA run number of shots iterations to geometrically optimize
%the shot position
h = waitbar(0,'Performing geometric optimization. Please wait...');
count = 1;
for i = 1:method(3)

    %Determine the helmet to optimize based on the remaining volume
    VolRemain = sum(sum(sum(StructMatGeo)));
    VolRemainTV = RatioTV * VolRemain;

    if method(4) == 0
        if VolRemainTV >= HelmetVolumes(4)
            Helmet(i) = 18;
        elseif VolRemainTV < HelmetVolumes(4) && VolRemainTV >=
HelmetVolumes(3)
            Helmet(i) = 14;
        elseif VolRemainTV < HelmetVolumes(2) && VolRemainTV >=
HelmetVolumes(2)
            Helmet(i) = 8;
        else
            Helmet(i) = 4;
        end
    else
        if VolRemainTV >= HelmetVolumes(4) && method(4) >= 18
            Helmet(i) = 18;
        elseif VolRemainTV >= HelmetVolumes(3) && method(4) >= 14
            Helmet(i) = 14;
        elseif VolRemainTV >= HelmetVolumes(2) && method(4) >= 8
            Helmet(i) = 8;
        else
            Helmet(i) = 4;
        end
    end
end

```

```

        end
    end

    %Alter the GA parameters based on the iteration
    if i <= 1
        PopulationSize_Data = 9;
        Generations_Data = 12;
        options = gaoptimset(options, 'PopulationSize',
PopulationSize_Data);
        options = gaoptimset(options, 'Generations', Generations_Data);
    elseif i <=3
        PopulationSize_Data = 7;
        Generations_Data = 10;
        options = gaoptimset(options, 'PopulationSize',
PopulationSize_Data);
        options = gaoptimset(options, 'Generations', Generations_Data);
    elseif i <=5
        PopulationSize_Data = 6;
        Generations_Data = 8;
        options = gaoptimset(options, 'PopulationSize',
PopulationSize_Data);
        options = gaoptimset(options, 'Generations', Generations_Data);
    elseif i <=8
        PopulationSize_Data = 5;
        Generations_Data = 6;
        options = gaoptimset(options, 'PopulationSize',
PopulationSize_Data);
        options = gaoptimset(options, 'Generations', Generations_Data);
    else
        PopulationSize_Data = 4;
        Generations_Data = 4;
        options = gaoptimset(options, 'PopulationSize',
PopulationSize_Data);
        options = gaoptimset(options, 'Generations', Generations_Data);
    end

    %Run the actual Genetic Algorithm Optimization
    [x,fval,exitflag,output,population,score] = ...

ga (@ (para) GeoGAOptFunc (para, StructMatGeo, MatIndX, MatIndY, MatIndZ, gamma,
Profiles, RxIsodose, Helmet(i), VolInit), nvars, [], [], [], [], lb, ub, [], option
s);

    %Convert the result of the GA optimization into the shot data form
    x = round(x*20)/20;
    shots = [Helmet(i) x 1];

    %Calculated the StructMatGeo matrix with the optimized shots
    %position to reflect the coverage
    [StructMatGeo] =
StructMatGeometricCalc (StructMatGeo, MatIndX, MatIndY, MatIndZ, shots, gamma,
Profiles, RxIsodose);
    %Uncomment here to show progress    %a =
StructMatGeo(:, :, 60);;figure(i+30);imagesc(a);
    %Add the shot to the created initial population
    X0Geo(1, count+2:count+5) = shots(1, 2:5);

```

```

    if isequal(shots(1),4)
        X0Geo(1,count:count+1) = [0 0];
    elseif isequal(shots(1),8)
        X0Geo(1,count:count+1) = [0 1];
    elseif isequal(shots(1),14)
        X0Geo(1,count:count+1) = [1 0];
    elseif isequal(shots(1),18)
        X0Geo(1,count:count+1) = [1 1];
    else
        error
    end
    count = count + 6;
    waitbar(i / method(3));
end
close(h);

```

getVolumePoints

```
function [x1V, y1V, z1V] = getVolumePoints(structNum, doseNum, planC)
%"getVolumePoint"
% Returns vectors for a specified structure and dose set, where
% the vector represents the x,y,z coordinate of the volume
% This function was modified from CERR getDVH function.
%
%
% [dosesV, volsV] = getDVH(structNum, doseNum, planC)

indexS = planC{end};
optS = planC{indexS.CERROptions};

%Get the scan number associated with the requested structure.
[scanSet, relStructNum] = getStructureAssociatedScan(structNum, planC);

ROIImageSize =
[planC{indexS.scan}(scanSet).scanInfo(1).sizeOfDimension1
planC{indexS.scan}(scanSet).scanInfo(1).sizeOfDimension2];

deltaY = planC{indexS.scan}(scanSet).scanInfo(1).grid1Units;

%Get raster segments for structure.
[segmentsM, planC, isError] = getRasterSegments(structNum, planC);

if isempty(segmentsM)
    isError = 1;
end
numSegs = size(segmentsM,1);

%Relative sampling of ROI voxels in this place, compared to CT spacing.
%Set when rasterSegments are generated (usually on import).
sampleRate = optS.ROISampleRate;

%Sample the rows
indFullV = 1 : numSegs;
if sampleRate ~= 1
    rV = 1 : length(indFullV);
    rV([rem(rV+sampleRate-1,sampleRate)~=0]) = [];
    indFullV = rV;
end

%Block process to avoid swamping on large structures
if isfield(optS, 'DVHBlockSize') & ~isempty(optS.DVHBlockSize)
    DVHBlockSize = optS.DVHBlockSize;
else
    DVHBlockSize = 5000;
end

blocks = ceil(length(indFullV)/DVHBlockSize);
volsV = [];
dosesV = [];
```

```

start = 1;

for b = 1 : blocks

    %Build the interpolation points matrix

    dummy = zeros(1,DVHBlockSize * ROIImageSize(1));
    x1V = dummy;
    y1V = dummy;
    z1V = dummy;
    volsSectionV = dummy;

    if start+DVHBlockSize > length(indFullV)
        stop = length(indFullV);
    else
        stop = start + DVHBlockSize - 1;
    end

    indV = indFullV(start:stop);

    mark = 1;
    for i = indV

        tmpV = segmentsM(i,1:10);
        delta = tmpV(5) * sampleRate;
        xV = tmpV(3): delta : tmpV(4);
        len = length(xV);
        rangeV = ones(1,len);
        yV = tmpV(2) * rangeV;
        zV = tmpV(1) * rangeV;
        sliceThickness = tmpV(10);
        %v = delta^2 * sliceThickness;
        v = delta * (deltaY*sampleRate) * sliceThickness;
        x1V(mark : mark + len - 1) = xV;
        y1V(mark : mark + len - 1) = yV;
        z1V(mark : mark + len - 1) = zV;
        volsSectionV(mark : mark + len - 1) = v;
        mark = mark + len;

    end

    %cut unused matrix elements
    x1V = x1V(1:mark-1);
    y1V = y1V(1:mark-1);
    z1V = z1V(1:mark-1);
    volsSectionV = volsSectionV(1:mark-1);

    %Get transformation matrices for both dose and structure.
    transMDose = getTransM('dose', doseNum, planC);
    transMStruct = getTransM('struct', structNum, planC);

    %Forward transform the structure's coordinates.
    if ~isempty(transMStruct)
        [x1V, y1V, z1V] = applyTransM(transMStruct, x1V, y1V, z1V);
    end
end

```

```

%Back transform the coordinates into the doses' coordinate system.
if ~isempty(transMDose)
    [x1V, y1V, z1V] = applyTransM(inv(transMDose), x1V, y1V, z1V);
end

start = stop + 1;

end

```


ImportLeksellPlan

```
function [planC, leksellInfo] = importLeksellPlan(planDir)
%"importLeksellPlan"
%   Imports a Leksell gamma knife plan into CERR format.  The directory
%   structure of the Leksell plan must be intact.
%
%By JRA 06/24/05
%
%LM: KRK, 05/31/07, all readLeksell<...>File functions are now complete
and
%           included in the importLeksellPlan function
%   KRK, 06/01/07, corrected the transform matrices of the scans
%   KRK, 06/04/07, added missing code in the structure import section
to
%           account for non associated scans
%   KRK, 06/05/07, added a second output variable (leksellInfo) to
hold all
%           information from the plan directory for easy access
into
%           the variables not currently used in CERR (metadata)
%   KRK, 06/07/07, the code for converting the normalized percentages
in
%           the dose matrices to units of Gy is now complete
%   KRK, 06/08/07, rescaled the imported scan images' intensities (MR
only)
%           so that they can now be viewed with the CERR CT
Window
%           preset "Head"
%   DK , 07/01/08, Rewrote the way directories are read. Change in
file
%           names and missing directories causing errors. Typo
%           errors in converting mm to cm. Also typo error in
%           reading CT Loop.
%
% copyright (c) 2001-2006, Washington University in St. Louis.
% Permission is granted to use or modify only for non-commercial,
% non-treatment-decision applications, and further only if this header
is
% not removed from any file. No warranty is expressed or implied for
any
% use whatever: use at your own risk. Commercial users can
% request a license.  Contact Joe Deasy for more information
% (radonc.wustl.edu@jdeasy, reversed). Users can request use of CERR
for
% institutional review board-approved protocols.
%
%
%Usage:
%   planC = importLeksellPlan(planDir)

LekC = [];

files = dir(planDir);
lekSellDirStr = {files.name};
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Importing directories
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1: length(lekSellDirStr)
    if files(i).isdir
        if strfind(upper(lekSellDirStr{i}), 'MR')
            LekC = readLeksellDir(planDir, 'MR', LekC,
lekSellDirStr{i});
        elseif strfind(lekSellDirStr{i}, 'CT')
            LekC = readLeksellDir(planDir, 'CT', LekC,
lekSellDirStr{i});
        else
            % try
            LekC = readLeksellDir(planDir, lekSellDirStr{i}, LekC,
lekSellDirStr{i});
            % catch
            % disp(['No Import Options for '
lekSellDirStr{i} ' ...'])
            % continue
            % end
        end
    end
end
end

%-----Initialize planC-----
---%
planC = initializeCERR;
indexS = planC{end};
planC{indexS.CERROptions} = CERROptions;

%-----Initialize scan information-----
---%
if isfield(LekC, 'mrScanS')
    for i=1:length(LekC.mrScanS)
        planC{indexS.scan}(i).scanType =
LekC.mrScanS(i).scanInfo(1).scanType;
        bool = ismember({LekC.studyS.modality},
LekC.mrScanS(i).scanInfo(1).scanType);
        matches = find(bool);
        if isempty(matches)
            planC{indexS.scan}(i).transM = [];
        else
            planC{indexS.scan}(i).transM =
LekC.studyS(matches(1)).rcsToxyzTransM;
        end

        planC{indexS.scan}(i).scanArray = LekC.mrScanS(i).scanArray;
        planC{indexS.scan}(i).scanInfo = LekC.mrScanS(i).scanInfo;

        planC{indexS.scan}(i).scanUID = createUID('scan');
    end
end
end

```

```

if isfield(LekC, 'ctScanS')

    for i = 1:length(LekC.ctScanS)
        indX = length(planC{indexS.scan}) + 1;
        planC{indexS.scan}(indX).scanType =
LekC.ctScanS(i).scanInfo(1).scanType;

        bool = ismember({LekC.studyS.modality},
LekC.ctScanS(i).scanInfo(1).scanType);
        matches = find(bool);
        planC{indexS.scan}(indX).transM =
LekC.studyS(matches(1)).rcsToxyzTransM;

        planC{indexS.scan}(indX).scanArray = LekC.ctScanS(i).scanArray;
        planC{indexS.scan}(indX).scanInfo = LekC.ctScanS(i).scanInfo;

        planC{indexS.scan}(indX).scanUID = createUID('scan');
    end
end

%Set the uniformized scan data in the plan (planC{indexS.scan})
planC = setUniformizedData(planC);
%The original transformation matrices read in by the readLeksellStudy
%function contained scaling as well as transformation/rotation. Since
CERR
%is compatible with only transformation/rotation matrices and scales
%according to its 3 voxel sizes, the scaling needs to be removed from
the
%current transformation matrices. To do this, multiply the original
transM
%by the inverse of the pure scaling matrix (multiplied by 10 since the
%voxel sizes are in cm and must be mm).
for i=1:length(planC{indexS.scan})
    if ~isempty(planC{indexS.scan}(i).transM)
        planC{indexS.scan}(i).transM = planC{indexS.scan}(i).transM /
[planC{indexS.scan}(i).uniformScanInfo.grid1Units*10 0 0 0; 0
planC{indexS.scan}(i).uniformScanInfo.grid2Units*10 0 0; 0 0
planC{indexS.scan}(i).uniformScanInfo.sliceThickness*10 0; 0 0 0 1];
        planC{indexS.scan}(i).transM(3,4) =
planC{indexS.scan}(i).transM(3,4) -
planC{indexS.scan}(i).uniformScanInfo.firstZValue;
    end
end

%-----Load structure(s)-----
---%
structNum = 1;
nonAssocCounter = 0;
for i=1:length(LekC.vols)
    %Check if the scan is either MR or CT, if not, disregard
    regTo = LekC.vols(i).registeredTo;
    bool = ismember({planC{indexS.scan}.scanType}, regTo);
    associatedScan = find(bool);
    if isempty(associatedScan)

```

```

        %Continue if not associated with any scan, eg Leksell Skull.
        nonAssocCounter = nonAssocCounter + 1; %keep a count of the non
associated scans to index associated scans correctly
        continue;
    end

    %Check for contours done on fused images (slicenums are therefore
out of
    %the range CT and MR images). There may be a way to map these
fusion
    %contours to MR or CT, possibly using the information from
    %LekC.fusionS, but it hasn't been completed yet.
    fusionContour = 0;
    for j=1:length(LekC.volS(i).contour)
        for k=1:length(planC{indexS.scan})
            if(LekC.volS(i).contour(j).sliceNum >
length(planC{indexS.scan}(k).scanInfo))
                fusionContour = 1;
                break;
            end
        end
    end
    if(fusionContour ~= 0)
        nonAssocCounter = nonAssocCounter + 1;
        continue;
    end

    %After checking if its associated, put in the structure information
    newStructS = newCERRStructure(associatedScan, planC);
    planC{indexS.structures} =
dissimilarInsert(planC{indexS.structures},newStructS,i -
nonAssocCounter);
    planC{indexS.structures}(i - nonAssocCounter).structureName =
LekC.volS(i).structName;
    contour = LekC.volS(i).contour;
    for j=1:length(contour)
        sliceNum = contour(j).sliceNum + 1; %+ 1 to match Matlab's
indexing (1-n+1 rather than 0-n)
        %scale the xy points based on the voxel sizes
        xyPoints = contour(j).contour;
        xyPoints(:,1) =
planC{indexS.scan}(associatedScan).uniformScanInfo.grid1Units *
contour(j).contour(:,1);
        xyPoints(:,2) =
planC{indexS.scan}(associatedScan).uniformScanInfo.grid2Units *
contour(j).contour(:,2);
        %set z values for the current x,y vertices
        xyPoints = [xyPoints
ones(size(xyPoints(:,1)))*planC{indexS.scan}(associatedScan).scanInfo(s
liceNum).zValue];
        %store the xyz points of each vertex
        planC{indexS.structures}(i -
nonAssocCounter).contour(sliceNum).segments.points = xyPoints;
    end
    structNum = structNum + 1; %keep a count of all of the associated
structures

```

```

    planC{indexS.structures}(i - nonAssocCounter).strUID =
createUID('structure');
    planC{indexS.structures}(i - nonAssocCounter).assocScanUID =
planC{indexS.scan}(associatedScan).scanUID;

    %Remove the contours from the second return variable to save space.
    %Only do this if the contour is read into CERR. In other words,
don't
    %remove the contours from this return variable if they are from a
non
    %associated scan (any contour not done on MR or CT, ex: fused).
    LekC.volS(i).contour = [];
end
%Calculate how to display the contoured segments inside of CERR
planC = getRasterSegs(planC);

%-----Load dose information-----
---%
normalizingConst = -1; %used to find the global maximum normalizing
constant in the dose matrix (see readLeksellDoseGroupFile for more
information)
%Put dose info into the CERR data structure.

%Set the Rx for the new dose grid
reply = input('Treatment today? ', 's');
matstring = strmatch(reply, strvcat('Yes', 'Y', 'yes', 'YES'));
if ~isempty(matstring)
    %doserate = input('Enter the dose rate for the imported plan
(Gy/min): ');
    doserate = 1;
    txdate = date;
else
    doserate = input('Enter the dose rate for the imported plan
(Gy/min): ');
    txdate = input('Enter the date of treatment (01-Jan-2010): ', 's');
end

if isfield(LekC, 'shotS')
    for i=1:length(LekC.shotS.doseMatrixInfo)
        dMI = LekC.shotS.doseMatrixInfo(i);

        planC{indexS.dose}(i).fractionGroupID = dMI.name;
        planC{indexS.dose}(i).horizontalGridInterval = dMI.gridSize;
        planC{indexS.dose}(i).verticalGridInterval = dMI.gridSize;
        planC{indexS.dose}(i).depthGridInterval = dMI.gridSize;

        try
            planC{indexS.dose}(i).doseArray = LekC.doses{i};
        catch
            planC{indexS.dose}(i) = [];
            continue
        end
    end
end

```

```

        % Temporarily hold the max of each dose matrix to see if it is
larger
        % than the other dose matrices' max value. If it is, store the
new
        % larger value in normalizingConst (this only affects plans
with multiple DMs).
        tmpNorm = max(planC{indexS.dose}(i).doseArray(:));
        if(tmpNorm > normalizingConst)
            normalizingConst = tmpNorm;
        end

        planC{indexS.dose}(i).coord1OFFirstPoint =
LekC.shotS.doseMatrixInfo(i).minExtent(1);
        planC{indexS.dose}(i).coord2OFFirstPoint =
LekC.shotS.doseMatrixInfo(i).minExtent(2);
        planC{indexS.dose}(i).coord3OFFirstPoint =
LekC.shotS.doseMatrixInfo(i).minExtent(3);

        siz = size(planC{indexS.dose}(i).doseArray);
        planC{indexS.dose}(i).sizeOfDimension1 = siz(2);
        planC{indexS.dose}(i).sizeOfDimension2 = siz(1);
        planC{indexS.dose}(i).sizeOfDimension3 = siz(3);

        planC{indexS.dose}(i).zValues =
linspace(LekC.shotS.doseMatrixInfo(i).minExtent(3),
LekC.shotS.doseMatrixInfo(i).maxExtent(3), siz(3));

        planC{indexS.dose}(i).doseUID = createUID('dose');
        planC{indexS.dose}(i).doseUnits = 'Gy';
        planC{indexS.dose}(i).assocScanUID = [];
        planC{indexS.dose}(i).doserate = doserate;
        planC{indexS.dose}(i).txdate = txdate;
    end
end
%Normalize the dose arrays and convert them to Gy using previously
found max.
%This converts all dose matrix values into CERR's native radiation
units, Gy.
for i=1:length(planC{indexS.dose})
    planC{indexS.dose}(i).doseArray =
(LekC.superS.globalMaxDose)*(planC{indexS.dose}(i).doseArray)/normalizi
ngConst;
    planC{indexS.dose}(i).Rx =
max(max(planC{indexS.dose}(i).doseArray));
end

%-----Load shot(s)--Written by Joseph-----
%-----%
if isfield(LekC,'shotS')
    disp('Importing Radiation Shot info...')
    for i=1:length(LekC.shotS.shots);
        dMI = LekC.shotS.shots(1,i);
        planC{indexS.beamGeometry}(1).shots(i).fractionGroupID =
LekC.shotS.doseMatrixInfo(1).name;
    end
end

```

```

        planC{indexS.beamGeometry}(1).shots(i).associatedMatrix =
dMI.associatedMatrix;
        planC{indexS.beamGeometry}(1).shots(i).targetpointLeksell =
dMI.targetPoint;
        planC{indexS.beamGeometry}(1).shots(i).targetpointTOGA =
dMI.targetPoint;
        planC{indexS.beamGeometry}(1).shots(i).x = dMI.targetPoint(1);
        planC{indexS.beamGeometry}(1).shots(i).y = dMI.targetPoint(2);
        planC{indexS.beamGeometry}(1).shots(i).z = dMI.targetPoint(3);
        planC{indexS.beamGeometry}(1).shots(i).weight = dMI.weight;
        planC{indexS.beamGeometry}(1).shots(i).gammaLeksell =
dMI.gamma;
        planC{indexS.beamGeometry}(1).shots(i).gammaTOGA = dMI.gamma;
        planC{indexS.beamGeometry}(1).shots(i).collimator_helmet =
dMI.collimator_helmet;
        planC{indexS.beamGeometry}(1).shots(i).plugging = dMI.plugging;
        planC{indexS.beamGeometry}(1).shots(i).srcWeight =
dMI.srcWeight;
        planC{indexS.beamGeometry}(1).shots(i).xyzSrcPositions =
dMI.xyzSrcPositions;
        planC{indexS.beamGeometry}(1).shots(i).skull =
dMI.xyzSrcPositions;
        planC{indexS.beamGeometry}(1).shots(i).time = 0;
    end
end

%-----Load skull info--Written by Joseph-----
%-----%
if isfield(LekC, 'skulls')
    planC{indexS.comment} = [];
    disp('Importing Skull info...')
    %Add skull measurement and angles info to planC.comment
    planC{indexS.comment}.skull = LekC.skulls;
    %Add skull contour info to planC.comment
    for i=1:length(LekC.volS);
        structName = LekC.volS(i).structName;
        if strcmp(structName, 'Skull_LGP')
            planC{indexS.comment}.skull.vol = LekC.volS(i).contour;
        end
    end
end

%-----Load frame info--Written by Joseph-----
%-----%
if isfield(LekC, 'skulls')
    disp('Importing Frame info...')
    planC{indexS.comment}.frame = LekC.targetframes;
end

%Log end of import

%-----Store metadata-----
%-----%

```

```

%Put all information obtained from the readLeksell<...>File functions
into
%the other return variable to view and/or possibly incorporate into
CERR.
%Also, get rid of some of the larger pieces of data (doses, pixel
arrays)
%that have already been read into the CERR variable to save space. See
the
%"Load Structures" section of this function to see how the contours are
%removed from this variable.

```

```

LekC.doses = [];
try
    for i=1:length(LekC.ctScanS)
        LekC.ctScanS(i).scanArray = [];
    end
end
try
    for i=1:length(LekC.mrScanS)
        LekC.mrScanS(i).scanArray = [];
    end
end
leksellInfo = LekC;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% READ THE DIRECTORY STRUCTURE OF LEKSELL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

```

```

function LekC = readLeksellDir(planDir,dirName, LekC, mrctdir)

```

```

switch dirName

```

```

    case 'Fusions'
        %Get Fusion info. (not used in CERR currently)
        disp('Reading Fusion info...')
        % fusionsFilename = fullfile(planDir, 'Fusions', 'Fusion.1');
        fusionsFilename = getLeksellFilesNames(fullfile(planDir,
'Fusions'), 'Fusion');
        for i = 1:length(fusionsFilename)
            LekC(i).fusionS = readLeksellFusionFile(fullfile(planDir,
'Fusions', fusionsFilename{i}));
        end

```

```

    case 'Orientations'
        %Get Orientations info. (not used in CERR currently)
        disp('Reading Orientations info...')

```



```

        % oriFilename = fullfile(planDir, 'Orientations',
'Orientation.1');
        oriFilename = getLeksellFilesNames(fullfile(planDir,
'Orientations'), 'Orientation');
        for i = 1:length(oriFilename)
            LekC(i).oriS = readLeksellOrientationFile(fullfile(planDir,
'Orientations', oriFilename{i}));
        end

        case 'SRdata'
            %Get Dosegroup data.
            disp('Reading Dosegroup info...')
            % doseFilename = fullfile(planDir, 'SRdata', ['DoseGroup'
num2str(doseNum) '.1']);

            doseFilename = getLeksellFilesNames(fullfile(planDir,
'SRdata'), 'DoseGroup');

            for i = 1:length(doseFilename)
                LekC.doses{i} = readLeksellDoseGroupFile(fullfile(planDir,
'SRdata', doseFilename{i}));
            end

            %Get skull info. (not used in CERR currently)
            disp('Reading Skull info...')
            % skullFilename = fullfile(planDir, 'SRdata', 'Skull.1');
            skullFilename = getLeksellFilesNames(fullfile(planDir,
'SRdata'), 'Skull');
            for i = 1:length(skullFilename)
                LekC(i).skullS = readLeksellSkullFile(fullfile(planDir,
'SRdata', skullFilename{i}));
            end

            %Get super info.
            disp('Reading Super info...')

            % superFilename = fullfile(planDir, 'SRdata', 'Super.1');
            superFilename = getLeksellFilesNames(fullfile(planDir,
'SRdata'), 'Super');
            for i = 1:length(superFilename )
                LekC(i).superS = readLeksellSuperFile(fullfile(planDir,
'SRdata', superFilename{i}));
            end

            %Get target frame info. (not used in CERR currently)
            disp('Reading TargetFrame info...')
            % targetframeFilename = fullfile(planDir, 'SRdata',
'TargetFrame.1');
            targetframeFilename = getLeksellFilesNames(fullfile(planDir,
'SRdata'), 'TargetFrame');
            for i = 1:length(targetframeFilename)
                LekC(i).targetframes =
readLeksellTargetframeFile(fullfile(planDir, 'SRdata',
targetframeFilename{i}));
            end

```

```

case 'Shots'
    %Get the shot info.
    disp('Reading Shot info...')

    % shotFilename = fullfile(planDir, 'Shots', 'Shot.1');
    shotFilename = getLeksellFilesNames(fullfile(planDir, 'Shots'),
'Shot');

    if isempty(shotFilename)
        shotFilename = getLeksellFilesNames(fullfile(planDir,
'Shots'), 'Group');
    end

    for i = 1:length(shotFilename)
        LekC(i).shotS = readLeksellShotsFile(fullfile(planDir,
'Shots', shotFilename{i}));
    end

case 'Studyregs'
    %Get the studyReg info. (not used in CERR currently)
    disp('Reading StudyReg info...')
    % srFilename = fullfile(planDir, 'Studyregs', 'Studyreg.1');
    srFilename = getLeksellFilesNames(fullfile(planDir,
'Studyregs'), 'studyreg');
    for i = 1:length(srFilename)
        LekC(i).studyRegS =
readLeksellStudyregFile(fullfile(planDir, 'Studyregs', srFilename{i}));
    end

case 'Studys'
    %Get the study info.
    disp('Reading Study info...')
    % studyFilename = fullfile(planDir, 'Studys', 'Study.1');
    studyFilename = getLeksellFilesNames(fullfile(planDir,
'Studys'), 'Study');
    for i = 1:length(studyFilename)
        LekC(i).studyS = readLeksellStudyFile(fullfile(planDir,
'Studys', studyFilename{i}));
    end

case 'Volumes'
    %Get volume info.
    disp('Reading Volume info...')
    % volFilename = fullfile(planDir, 'Volumes', 'Volume.1');
    volFilename = getLeksellFilesNames(fullfile(planDir,
'Volumes'), 'Volume');
    for i = 1:length(volFilename)
        LekC(i).volS = readLeksellVolumeFile(fullfile(planDir,
'Volumes', volFilename{i}));
    end

case 'Plugs'
    %Get plug info. (not used in CERR currently)

```

```

disp('Reading Plug info...')

% plugFilename = fullfile(planDir, 'Plugs', 'Plug.1');
plugFilename = getLeksellFilesNames(fullfile(planDir, 'Plugs'),
'Plug');
for i = 1:length(plugFilename)
    LekC(i).plugS = readLeksellPlugFile(fullfile(planDir,
'Plugs', plugFilename{i}));
end

case 'Sortings'
    %Get sorting info. (not used in CERR currently)
    disp('Reading Sorting info...')
    % sortingFilename = fullfile(planDir, 'Sortings', 'Sorting.1');
    sortingFilename = getLeksellFilesNames(fullfile(planDir,
'Sortings'), 'Sorting');
    for i = 1:length(sortingFilename)
        LekC(i).sortingS = readLeksellSortingFile(fullfile(planDir,
'Sortings', sortingFilename{i}));
    end

case 'Thresholds'
    %Get threshold info. (not used in CERR currently)
    disp('Reading Threshold info...')
    % thresholdFilename = fullfile(planDir, 'Thresholds',
'Threshold.1');
    thresholdFilename = getLeksellFilesNames(fullfile(planDir,
'Thresholds'), 'Threshold');
    for i = 1:length(thresholdFilename)
        LekC(i).thresholdS =
readLeksellThresholdFile(fullfile(planDir, 'Thresholds',
thresholdFilename{i}));
    end

case 'MR'
    disp('Reading MR scan info...')

    if isfield(LekC, 'mrScanS')
        indX = length(LekC.mrScanS) + 1;
    else
        indX = 1;
    end

    imageDir = fullfile(planDir, mrctdir);
    [mat, scanInfo] = readLeksellImageset(imageDir);
    if isempty(mat)
        return
    end
    LekC.mrScanS(indX).scanArray = mat;
    LekC.mrScanS(indX).scanInfo = scanInfo;
    %Put the pixel intensities in the range of 1000-1700. This code
works, but
    %is not generally correct. Modify the commented code under
this to correct
    %the MR pixel intensity problem.

```

```

        LekC.mrScanS(indX).scanArray = LekC.mrScanS(indX).scanArray -
median(LekC.mrScanS(indX).scanArray(:)) + 1000;
        %Match the scan to the study, then subtract the lower of the
ranges to
        %standardize the pixel intensity ranges (make it go from 0-722
rather than
        %its original values in the range of 32000-33000).
        %   for j=1:length(LekC.studyS)
        %       if(strmatch(LekC.studyS(j).modality,
LekC.mrScanS(indX).scanInfo(1).scanType))
        %           LekC.mrScanS(indX).scanArray =
LekC.mrScanS(indX).scanArray -
        %               LekC.studyS(j).pixelIntensityRange(1);
        %               break;
        %       end
        %   end

case 'CT'
    if isfield(LekC,'ctScanS')
        indX = length(LekC.ctScanS) + 1;
    else
        indX = 1;
    end
    disp('Reading CT scan info...')

    imageDir = fullfile(planDir, mrctdir);

    [mat, scanInfo] = readLeksellImageset(imageDir);
    if isempty(mat)
        return
    end

    LekC.ctScanS(indX).scanInfo = scanInfo;

    LekC.ctScanS(indX).scanArray = mat;
end

```

ImportCurrentPlan2X0PlanC

```
function [planC] = ImportCurrentPlan2X0PlanC(planC, doseindex,
structindex);
%
%function [planC] = ImportCurrentPlan2X0PlanC(planC, doseindex,
structindex);
%
%ImportCurrentPlan2X0PlanC - Function that adds the current plan to the
X0
%variable in planC
%
%doseindex - Plan index to import X0 into
%X0Index - index of X0 plan to import to planC
%gamma - gamma angle of shots
%
%Joseph Bowling University of TN 12/14/2011

%Setup the problem
indexS = planC{end};

%Get the current plan
shots = planC{indexS.beamGeometry}(doseindex).shots;
for i = 1:length(shots)
    X0Exist(i,3) = shots(i).x;
    X0Exist(i,4) = shots(i).y;
    X0Exist(i,5) = shots(i).z;
    X0Exist(i,6) = shots(i).weight;
    X0Exist(i,1) = shots(i).collimator_helmet;
end
for i = 1:length(shots)
    if isequal(X0Exist(i,1),4)
        X0Exist(i,1:2) = [0 0];
    elseif isequal(X0Exist(i,1),8)
        X0Exist(i,1:2) = [0 1];
    elseif isequal(X0Exist(i,1),14)
        X0Exist(i,1:2) = [1 0];
    elseif isequal(X0Exist(i,1),18)
        X0Exist(i,1:2) = [1 1];
    else
        error
    end
end

%Reshape X0Exist variable
X0Existsize = size(X0Exist);
X0Exist = reshape(X0Exist',1,X0Existsize(1)*X0Existsize(2));

%Add E0Exist to the X0 variable
X0 = planC{indexS.comment}.GAOptInitPopulation;
numplans = size(X0,1);
X0{numplans+1,:} = X0Exist;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
```

```

%Alter X0 so that each plan is the same length. This involves adding
dummy
%4 mm shots with zero weight. These dummy four shots will be added
%homogeneously throughout the ROI to be optimized
[x1V, y1V, z1V] = getVolumePoints(structindex, doseindex, planC);
VolPoints = [x1V' y1V' z1V'];
VolSpread = [min(VolPoints); max(VolPoints)];
for i = 1:length(X0)
    MaxNumShots(i) = length(X0{i}(:));
end
MaxNumShots = max(MaxNumShots);
for i = 1:length(X0)
    numshots2add(i) = (MaxNumShots - length(X0{i}(:))) / 6;
end
for i = 1:length(X0)
    if numshots2add(i) > 0
        ShotsAdd = zeros(numshots2add(i), 6);
        ShotsAdd(:, [3]) = VolSpread(1,1) + (VolSpread(2,1) -
VolSpread(1,1)).*rand(numshots2add(i),1);
        ShotsAdd(:, [4]) = VolSpread(1,2) + (VolSpread(2,2) -
VolSpread(1,2)).*rand(numshots2add(i),1);
        ShotsAdd(:, [5]) = VolSpread(1,3) + (VolSpread(2,3) -
VolSpread(1,3)).*rand(numshots2add(i),1);
        ShotsAdd =
reshape(ShotsAdd', 1, size(ShotsAdd,1)*size(ShotsAdd,2));
        X0{i} = [X0{i} ShotsAdd];
    end
end

%Assign created initial population to the planC variable in the comment
%section
planC{indexS.comment}.GAOptInitPopulation = X0;

%Update planC with the new dose matrix position
assignin('base', 'planC', planC);

```

ImportX0GUI

```
function varargout = ImportX0GUI(varargin)
% IMPORTX0GUI MATLAB code for ImportX0GUI.fig
%     IMPORTX0GUI, by itself, creates a new IMPORTX0GUI or raises the
existing
%     singleton*.
%
%     H = IMPORTX0GUI returns the handle to a new IMPORTX0GUI or the
handle to
%     the existing singleton*.
%
%     IMPORTX0GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in IMPORTX0GUI.M with the given input
arguments.
%
%     IMPORTX0GUI('Property','Value',...) creates a new IMPORTX0GUI or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before ImportX0GUI_OpeningFcn gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to ImportX0GUI_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ImportX0GUI

% Last Modified by GUIDE v2.5 13-Jun-2011 10:52:40

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @ImportX0GUI_OpeningFcn, ...
                  'gui_OutputFcn',  @ImportX0GUI_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

% End initialization code - DO NOT EDIT

% --- Executes just before ImportX0GUI is made visible.
function ImportX0GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to ImportX0GUI (see VARARGIN)

% Choose default command line output for ImportX0GUI
handles.output = hObject;

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
end

%Populate objects in the GUI
setGUIpopupmenuPlan(handles);
setGUIpopupmenuX0(handles);
set(handles.popupmenuGamma,'String',{'72' '90' '110' '125'});
set(handles.popupmenuGamma,'Value',2);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ImportX0GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ImportX0GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenuPlan.
function popupmenuPlan_Callback(hObject, eventdata, handles)

```



```

% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuPlan contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuPlan

% --- Executes during object creation, after setting all properties.
function popupmenuPlan_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuX0.
function popupmenuX0_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuX0 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenuX0
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuX0

% --- Executes during object creation, after setting all properties.
function popupmenuX0_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuX0 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

```

```

% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Get the parameters set for the optimization
doseindex = get(handles.popupmenuPlan, 'Value');
X0index = get(handles.popupmenuX0, 'Value');
gammaSel = get(handles.popupmenuGamma, 'Value');
switch gammaSel
    case 1
        gamma = 72;
    case 2
        gamma = 90;
    case 3
        gamma = 110;
    case 4
        gamma = 125;
end

[handles.planC] = ImportX0toPlanC(handles.planC, doseindex, X0index,
gamma);

%Get the selected plan
planindex = doseindex;

%Get the calculation method
CalcMethod = 1;

[Profiles] = loadprofilesTOGA(handles.planC, planindex);

%Calculate the dose
[handles.planC] =
dosecalcTOGA(handles.planC, planindex, Profiles, CalcMethod);

fprintf('Initial Plan: %d \n\n', X0index);

%Print the conformity indices
[Iso_Cov, Cov_RT0G, CI_RT0G, CI_Paddick, CI_Gradient, CI_Karolinska] =
ConformIndex(handles.planC, 1, planindex, 0.5)

%Update planC with the new dose matrix position
assignin('base', 'planC', handles.planC);
guidata(hObject, handles);

function setGUIpopupmenuPlan(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)

```

```

    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
end
set(handles.popupmenuPlan, 'String', plannames);

function setGUIpopupmenuX0(handles)
% handles    structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
X0size =
size(handles.planC{handles.indexS.comment}.GAOptInitPopulation,1);
for i = 1:X0size
    X0names{i,1} = num2str(i);
end
set(handles.popupmenuX0, 'String', X0names);

% --- Executes on selection change in popupmenuGamma.
function popupmenuGamma_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuGamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
% popupmenuGamma contents as cell array
%      contents{get(hObject,'Value')} returns selected item from
% popupmenuGamma

% --- Executes during object creation, after setting all properties.
function popupmenuGamma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuGamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called

% Hint: popupmenu controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

ImportX0toPlanC

```
function [planC] = ImportX0toPlanC(planC, doseindex, X0Index, gamma);
%
%function [planC] = ImportX0toPlanC(planC, doseindex, X0Index,
doseindex);
%
%ImportX0toPlanC - Function to import a Gamma Knife treatment plan
created
%by GAInitialPopulation into the planC{beamGeometry}(doseindex)
structure
%
%doseindex - Plan index to import X0 into
%X0Index - index of X0 plan to import to planC
%gamma - gamma angle of shots
%
%Joseph Bowling University of TN 5/11/2011

%Setup the problem
indexS = planC{end};

%Import X0 shots into planC
para = planC{indexS.comment}.GAOptInitPopulation(X0Index,:);
para = cell2mat(para);
para = reshape(para,6,length(para)/6)';
numshots = size(para);
shots = zeros(numshots(1),6);
for i = 1:numshots(1)
    if isequal(para(i,1:2),[0 0])
        unbincol(i) = 4;
    elseif isequal(para(i,1:2),[0 1])
        unbincol(i) = 8;
    elseif isequal(para(i,1:2),[1 0])
        unbincol(i) = 14;
    elseif isequal(para(i,1:2),[1 1])
        unbincol(i) = 18;
    else
        error
    end
end
shots(:,1) = unbincol';
shots(:,2) = para(:,3);
shots(:,3) = para(:,4);
shots(:,4) = para(:,5);
shots(:,5) = para(:,6);
shots(:,6) = gamma;
shots(:,7) = 0;

for i = 1:numshots
    shothold(1,i).collimator_helmet = shots(i,1);
    shothold(1,i).x = shots(i,2);
    shothold(1,i).y = shots(i,3);
    shothold(1,i).z = shots(i,4);
    shothold(1,i).weight = shots(i,5);
```

```

        shothold(1,i).gammaLeksell = shots(i,6);
        shothold(1,i).gammaTOGA = shots(i,6);
        shothold(1,i).fractionGroupID =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).fractionGroupID;
        shothold(1,i).associatedMatrix =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).associatedMatrix;
        shothold(1,i).targetpointLeksell =
[shots(i,2);shots(i,3);shots(i,4)];
%handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).targetpoint
Leksell;
        shothold(1,i).targetpointTOGA = [shots(i,2);shots(i,3);shots(i,4)];
%handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).targetpoint
TOGA;
        shothold(1,i).plugging =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).plugging;
        shothold(1,i).srcWeight =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).srcWeight;
        shothold(1,i).xyzSrcPositions =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).xyzSrcPositions;
        shothold(1,i).skull =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).skull;
        shothold(1,i).time =
planC{indexS.beamGeometry}(1,doseindex).shots(1,1).time;
end

%Input optimized shots to planC
planC{indexS.beamGeometry}(1,doseindex).shots = shothold;

```

loadProfilesTOGA

```
function [Profiles] = loadprofilesTOGA(planC,planindex,gamma);
%
%[Profiles] = loadprofilesTOGA(planC);
%
%Loads the necessary Gamma Knife helmet dose profiles based shots in
the
%selected plan (planindex) contained in the planC structure unless
variable
%gamma is provided, then function loades all available dose profiles
for angle gamma
%
%Joseph Bowling 10-10-10 revised 3-4-11

%Check to see if the Profiles variable already exist and determine what
%profiles have anyready been loaded
ProfExist = evalin('base','ismember({'Profiles'},who)');
if ProfExist
    Profiles = evalin('base','Profiles');
    if ~isempty(Profiles)
        names1 = fieldnames(Profiles);
    end
    numload = size(names1);
    count = 1;
    for i = 1:numload(1)
        temp = char(names1(i));
        temp2 = ['fieldnames(Profiles.', temp, ')'];
        names2 = eval(temp2);
        numload2 = size(names2);
        for j = 1:numload2(1)
            celltemp = names2(j);
            tempname = ['Profiles.', temp, '.', char(celltemp)];
            LoadedProfiles(count) = cellstr(tempname);
            count = count + 1;
        end
    end
else
    LoadedProfiles = {' '};
end

%Get the shots to determine what dose profiles need to be loaded
indexS = planC{end};
shots = planC{indexS.beamGeometry}(1,planindex).shots;

num = size(shots);
for i = 1:num(2)
    HelmetGammaUsed(i,1) = shots(i).collimator_helmet;
    HelmetGammaUsed(i,2) = shots(i).gammaTOGA;
end
HelmetGammaUsed = unique(HelmetGammaUsed,'rows');

%If Gamma Angle provide load all profiles
if nargin == 3
```

```

counter = 1;
for i = [4 8 14 18]
    NeededTemp = ['Profiles.H', num2str(i), '.G', num2str(gamma)];
    NeededProfiles(counter) = cellstr(NeededTemp);
    counter = counter + 1;
end
else
    num = size(HelmetGammaUsed);
    for i = 1:num(1)
        NeededTemp = ['Profiles.H', num2str(HelmetGammaUsed(i,1)),
'.G', num2str(HelmetGammaUsed(i,2))];
        NeededProfiles(i) = cellstr(NeededTemp);
    end
end

%Determine what shots need to loaded
Need2Load = ~ismember(NeededProfiles,LoadedProfiles);

%num = nnz(Need2Load);
num = size(Need2Load);
for i = 1:num(2)
    if Need2Load(i) == 1
        str = char(NeededProfiles(i));
        k = findstr(str, '.H');
        str = [str(1:k-2) str(k+2:end)];
        k = findstr(str, '.');
        str = [str(1:k-1), 'mm', str(k+1:end)];
        strcomputer = computer;
        if strcmp(strcomputer, 'PCWIN')
            loadstring = ['load(''C:\TOGA\DoseProfiles\'', str, '');"'];
        elseif strcmp(strcomputer, 'MACI64')
            loadstring = ['load(''/TOGA/DoseProfiles/'', str, '');"'];
        end
        evalin('base',loadstring);

        k = findstr(char(NeededProfiles(i)), '.H');
        temp = char(NeededProfiles(i));
        temp = temp(k+1:end);
        evalstr = [char(NeededProfiles(i)), ' = ', temp, ';'];
        evalin('base',evalstr);
    end
end

Profiles = evalin('base','Profiles;');

```

mutationgaussianTOGA

```
function mutationChildren =  
mutationgaussianTOGA(parents,options,GenomeLength,FitnessFcn,state,this  
Score,thisPopulation,scale,shrink)  
%MUTATIONGAUSSIAN Gaussian mutation.  
%   MUTATIONCHILDREN =  
MUTATIONGAUSSIAN(PARENTS,OPTIONS,GENOMELENGTH,...  
%   FITNESSFCN,STATE,THISSCORE,THISPOPULATION,SCALE,SHRINK) Creates the  
%   mutated children using the Gaussian distribution.  
%  
%   SCALE controls what fraction of the gene's range is searched. A  
%   value of 0 will result in no change, a SCALE of 1 will result in a  
%   distribution whose standard deviation is equal to the range of this  
gene.  
%   Intermediate values will produce ranges in between these extremes.  
%  
%   SHRINK controls how fast the SCALE is reduced as generations go by.  
%   A SHRINK value of 0 will result in no shrinkage, yielding a  
constant search  
%   size. A value of 1 will result in SCALE shrinking linearly to 0 as  
%   GA progresses to the number of generations specified by the options  
%   structure. (See 'Generations' in GAOPTIMSET for more details).  
Intermediate  
%   values of SHRINK will produce shrinkage between these extremes.  
%   Note: SHRINK may be outside the interval (0,1), but this is ill-  
advised.  
%  
%   Example:  
%       options = gaoptimset('MutationFcn',{@mutationgaussian});  
%  
%   This specifies that the mutation function used will be  
%   MUTATIONGAUSSIAN, and since no values for SCALE or SHRINK are  
specified  
%   the default values are used.  
%  
%       scale = 1; shrink = 1;  
%       options =  
gaoptimset('MutationFcn',{@mutationgaussian,scale,shrink});  
%  
%   This specifies that the mutation function used will be  
%   MUTATIONGAUSSIAN, and the values for SCALE or SHRINK are specified  
%   as 0.5 and 0.75 respectively.  
%  
%   Copyright 2003-2009 The MathWorks, Inc.  
%   $Revision: 1.1.6.2 $   $Date: 2009/10/10 20:08:27 $  
%  
% Use default parameters if the are not passed in.  
% If these defaults are not what you prefer, you can pass in your own  
% values when you set the mutation function:  
%  
% options.MutationFunction = { mutationgaussian, 0.3, 0} ;  
  
%savefile = 'mutationinput.mat';  
%save(savefile,'parents','options','GenomeLength','FitnessFcn',  
'state','thisScore','thisPopulation','scale','shrink');
```



```

numshots = GenomeLength/6;

if nargin < 9 || isempty(shrink)
    shrink = 1;
    if nargin < 8 || isempty(scale)
        scale = 1;
    end
end

if (shrink > 1) || (shrink < 0)
    msg = sprintf('Shrink factors that are less than zero or greater
than one may \n\t\t result in unexpected behavior. ');
    warning('globaloptim:mutationgaussian:shrinkFactor',msg);
end

%Mutate the double variable
scale = scale - shrink * scale * state.Generation/options.Generations;
range = options.PopInitRange;
lower = range(1,:);
upper = range(2,:);
scale = scale * (upper - lower);
mutationChildren = zeros(length(parents),GenomeLength);
for i=1:length(parents)
    parent = thisPopulation(parents(i),:);
    mutationChildren(i,:) = parent + scale .* randn(1,length(parent));
end

%Mutate the Binary Variables (This is uniform)
mutationRate = 0.01;
mutationChildrenBit = zeros(length(parents),GenomeLength);
for i=1:length(parents)
    child = thisPopulation(parents(i),:);
    mutationPoints = find(rand(1,length(child)) < mutationRate);
    child(mutationPoints) = ~child(mutationPoints);
    mutationChildrenBit(i,:) = child;
end

%Combine created Double and Bit mutation children
for i = 1:numshots
    count = [1+(i-1)*6 2+(i-1)*6];
    mutationChildren(:,count) = mutationChildrenBit(:,count);
end
%mutationChildren

```

newDoseGridTOGA

```
function [planC] =
newdosegridTOGA(planC,method,MatrixName,Rx,TxDate,TxDR,copyplan,planval
);
%
%[planC] = newdosegridTOGA(planC);
%
%Initializes a new dose grid
%
%planC - CERR plan variable
%method - Dose grid placement method (1-Graphical, 2-Shots, 3-
Replicate)
%MatrixName - The name of the dose matrix
%Rx - Radiation prescription max dose
%TxDate - Date of treatment
%TXDR - Doserate for date of treatment
%
%Joseph Bowling 10-17-10

%Setup of code and define the index
indexS = planC{end};
numslices = length(planC{indexS.scan}(1,1).scanInfo);
index = length(planC{indexS.dose}) + 1;

%Define the new dose matrix
planC{indexS.dose}(index).fractionGroupID = MatrixName;
planC{indexS.dose}(index).horizontalGridInterval = 0.05;
planC{indexS.dose}(index).verticalGridInterval = 0.05;
planC{indexS.dose}(index).depthGridInterval = 0.05;
planC{indexS.dose}(index).doseScale = 1;
planC{indexS.dose}(index).doseUnits = 'Gy';
planC{indexS.dose}(index).doserate = TxDR;
%planC{indexS.dose}(index).doserate = input('Enter the dose rate for
the imported plan (Gy/min): ');
if ~isempty(planC{indexS.scan}(1,1).transM)
    ztransformunknown = planC{indexS.scan}(1,1).transM(3,4);
else
    ztransformunknown = 0;
end
%Depending on the method selected, define location of new dose grid
switch method
case 1
    figure(2);
    corner1 = round(ginput(1));
    corner2 = round(ginput(1));
    minextent = min(corner1,corner2);
    maxextent = max(corner1,corner2);
    planC{indexS.dose}(index).coord1OFFFirstPoint = minextent(1);
    planC{indexS.dose}(index).coord2OFFFirstPoint = minextent(2);
    planC{indexS.dose}(index).coord3OFFFirstPoint =
planC{indexS.scan}(1,1).scanInfo(1,1).zValue + ztransformunknown;
    minextent(3) = planC{indexS.scan}(1,1).scanInfo(1,1).zValue +
ztransformunknown;
```

```

        maxextent(3) =
planC{indexS.scan}(1,1).scanInfo(1,length(planC{indexS.scan}(1,1).scanI
nfo)).zValue + ztransformunknown;
    case 2
        for i = 1:length(planC{indexS.beamGeometry})
            shots(i,:) = planC{indexS.beamGeometry}(1,i).shots;
        end
        for i = 1:length(shots)
            x(i) = shots(i).x;
            y(i) = shots(i).y;
            z(i) = shots(i).z;
        end
        minextent = [min(x) min(y) min(z)] - 2;
        maxextent = [max(x) max(y) max(z)] + 2;
        planC{indexS.dose}(index).coord1OFFirstPoint = minextent(1);
        planC{indexS.dose}(index).coord2OFFirstPoint = minextent(2);
        planC{indexS.dose}(index).coord3OFFirstPoint = minextent(3);
    case 3
        minextent = [planC{indexS.dose}(planval).coord1OFFirstPoint
planC{indexS.dose}(planval).coord2OFFirstPoint
planC{indexS.dose}(planval).coord3OFFirstPoint];
        gridsize = planC{indexS.dose}(planval).depthGridInterval;
        orggridsize = [planC{indexS.dose}(planval).sizeOfDimension2
planC{indexS.dose}(planval).sizeOfDimension1
planC{indexS.dose}(planval).sizeOfDimension3];
        gridsize = gridsize * orggridsize;
        maxextent = gridsize + minextent;
        planC{indexS.dose}(index).coord1OFFirstPoint =
planC{indexS.dose}(planval).coord1OFFirstPoint;
        planC{indexS.dose}(index).coord2OFFirstPoint =
planC{indexS.dose}(planval).coord2OFFirstPoint;
        planC{indexS.dose}(index).coord3OFFirstPoint =
planC{indexS.dose}(planval).coord3OFFirstPoint;
    end

%Create the dose matrix
dosedimen = (maxextent - minextent)/.05;
planC{indexS.dose}(index).doseArray =
zeros(dosedimen(2),dosedimen(1),dosedimen(3),'single');

%Not certain what this is
siz = size(planC{indexS.dose}(index).doseArray);
planC{indexS.dose}(index).sizeOfDimension1 = siz(2);
planC{indexS.dose}(index).sizeOfDimension2 = siz(1);
planC{indexS.dose}(index).sizeOfDimension3 = siz(3);
%minz = planC{indexS.scan}(1,1).scanInfo(1,1).zValue +
ztransformunknown;
%maxz =
planC{indexS.scan}(1,1).scanInfo(1,length(planC{indexS.scan}(1,1).scanI
nfo)).zValue + ztransformunknown;
planC{indexS.dose}(index).zValues =
linspace(minextent(3),maxextent(3),dosedimen(3));

%I can't remember why I did this, not in CERR import

```

```

zlength = 0.05 * siz(3) + planC{indexS.scan}(1,1).scanInfo(1,1).zValue;
xlength = 0.05 * siz(2) + minextent(1);
ylength = 0.05 * siz(1) + minextent(2);
planC{indexS.dose}(index).align.x = minextent(1):0.05:xlength-0.05;
planC{indexS.dose}(index).align.y = minextent(2):0.05:ylength-0.05;
%planC{indexS.dose}(index).align.z =
planC{indexS.scan}(1,1).scanInfo(1,1).zValue:0.05:zlength-0.05; changed
with new version of CERR
planC{indexS.dose}(index).align.z = minextent(3):0.05:maxextent(3);

%Set the Rx for the new dose grid
planC{indexS.dose}(index).Rx = Rx;
planC{indexS.dose}(index).txdate = TxDate;
planC{indexS.dose}(index).doseUID = planC{indexS.dose}(1).doseUID;

%Create the shots variable
if copyplan == 1
    planC{indexS.beamGeometry}(index).shots =
planC{indexS.beamGeometry}(planval).shots;
    for j = 1:length(planC{indexS.beamGeometry}(1,index).shots)
        planC{indexS.beamGeometry}(index).shots(j).fractionGroupID =
planC{indexS.dose}(index).fractionGroupID;
        planC{indexS.beamGeometry}(index).shots(j).associatedMatrix =
index - 1;
    end
else
    planC{indexS.beamGeometry}(index).shots =
planC{indexS.beamGeometry}(1).shots(1);
    planC{indexS.beamGeometry}(index).shots.fractionGroupID =
planC{indexS.dose}(index).fractionGroupID;
    planC{indexS.beamGeometry}(index).shots.associatedMatrix = index-1;
    planC{indexS.beamGeometry}(index).shots.targetpointLeksell = [];
    planC{indexS.beamGeometry}(index).shots.targetpointTOGA = [];
    planC{indexS.beamGeometry}(index).shots.x = [];
    planC{indexS.beamGeometry}(index).shots.y = [];
    planC{indexS.beamGeometry}(index).shots.z = [];
    planC{indexS.beamGeometry}(index).shots.weight = [];
    planC{indexS.beamGeometry}(index).shots.gammaLeksell = [];
    planC{indexS.beamGeometry}(index).shots.gammaTOGA = [];
    planC{indexS.beamGeometry}(index).shots.collimator_helmet = [];
    planC{indexS.beamGeometry}(index).shots.plugging = [];
end

```

OptGATOGAGUI

```
function varargout = OptGATOGAGUI(varargin)
% OPTGATOGAGUI M-file for OptGATOGAGUI.fig
%     OPTGATOGAGUI, by itself, creates a new OPTGATOGAGUI or raises
the existing
%     singleton*.
%
%     H = OPTGATOGAGUI returns the handle to a new OPTGATOGAGUI or the
handle to
%     the existing singleton*.
%
%     OPTGATOGAGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in OPTGATOGAGUI.M with the given input
arguments.
%
%     OPTGATOGAGUI('Property','Value',...) creates a new OPTGATOGAGUI
or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before OptGATOGAGUI_OpeningFcn gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to OptGATOGAGUI_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help OptGATOGAGUI

% Last Modified by GUIDE v2.5 11-Nov-2011 12:48:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @OptGATOGAGUI_OpeningFcn, ...
                  'gui_OutputFcn',  @OptGATOGAGUI_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

% End initialization code - DO NOT EDIT

% --- Executes just before OptGATOGAGUI is made visible.
function OptGATOGAGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to OptGATOGAGUI (see VARARGIN)

% Choose default command line output for OptGATOGAGUI
handles.output = hObject;

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
end

%Populate objects in the GUI
setGUIpopupmenuPlan(handles);
setGUIpopupmenuROI(handles);
setGUILabelSlideValue(handles);

set(handles.popupmenuGamma,'String',{'72' '90' '110' '125'});
set(handles.popupmenuGamma,'Value',2);

if
~isempty(handles.planC{handles.indexS.comment}(1,1).GAOptInitPopulation
)
    numshots =
length(handles.planC{handles.indexS.comment}(1,1).GAOptInitPopulation{1
,1}) / 6;
    numshotstext = num2str(numshots);
    set(handles.text23,'String',numshotstext);
end
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes OptGATOGAGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = OptGATOGAGUI_OutputFcn(hObject, eventdata,
handles)

```

```

% varargout    cell array for returning output args (see VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenuPlan.
function popupmenuPlan_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuPlan contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuPlan

% --- Executes during object creation, after setting all properties.
function popupmenuPlan_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit1_Callback(hObject, eventdata, handles)
% hObject      handle to edit1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%           str2double(get(hObject,'String')) returns contents of edit1 as
a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

```

```

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuStructure.
function popupmenuStructure_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuStructure (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuStructure contents as cell array
%          contents{get(hObject,'Value')} returns selected item from
popupmenuStructure

% --- Executes during object creation, after setting all properties.
function popupmenuStructure_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuStructure (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%          get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

setGUILabelSlideValue(handles);

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to slider1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

```



```

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function edit2_Callback(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%          str2double(get(hObject,'String')) returns contents of edit2 as
a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Get the parameters set for the optimization
warning off
indexS = handles.planC{end};
doseindex = get(handles.popupmenuPlan,'Value');
gammaSel = get(handles.popupmenuGamma,'Value');
switch gammaSel
    case 1
        gamma = 72;
    case 2
        gamma = 90;
    case 3
        gamma = 110;
    case 4
        gamma = 125;
end
[Profiles] = loadprofilesTOGA(handles.planC,doseindex,gamma);

```

```

structNum = get(handles.popupmenuStructure, 'Value');
if get(handles.checkbox1, 'Value');
    structNum(2,1) = get(handles.popupmenu7, 'Value');
end
if get(handles.checkbox2, 'Value');
    structNum(3,1) = get(handles.popupmenu8, 'Value');
    structNum(3,2) = str2num(get(handles.edit12, 'String'));
end

RxIsodose = get(handles.slider1, 'Value');
TV = getStructureVol(structNum(1), handles.planC);
CalcMethod = 1;
wstr = get(handles.edit2, 'String');
w = str2num(wstr);

%Define the optimization function
OptFunc =
@(para) TOGAGAOptFunc(para, handles.planC, doseindex, Profiles, gamma, w, structNum, RxIsodose, TV, CalcMethod);

%Determine the range of values for the parameters (ie spread of x,y,z
of
%the target volume) for the parameter optimization range.
numslices =
length(handles.planC{indexS.structures}(1, structNum(1)).contour);
count = 1;
for i = 1:numslices;
    if ~isempty(handles.planC{indexS.scan}(1,1).transM)
        if
~isempty(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).
segments.points)
            x(count,:) =
[ min(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).segments.points(:,1))
max(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).segments.points(:,1)) ];
            y(count,:) =
[ min(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).segments.points(:,2))
max(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).segments.points(:,2)) ];
            z(count,:) =
[ min(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).segments.points(:,3))
max(handles.planC{indexS.structures}(1, structNum(1)).contour(1,i).segments.points(:,3)) ];
            count = count + 1;
        end
    else
        if
~isempty(handles.planC{indexS.structures}(1, structNum(1)).contour(i).segments)
            x(count,:) =
[ min(handles.planC{indexS.structures}(1, structNum(1)).contour(i).segments(1).points(:,1))

```

```

max(handles.planC{indexS.structures}(1,structNum(1)).contour(i).segments(1).points(:,1));
    y(count,:) =
[min(handles.planC{indexS.structures}(1,structNum(1)).contour(i).segments(1).points(:,2))
max(handles.planC{indexS.structures}(1,structNum(1)).contour(i).segments(1).points(:,2))];
    z(count,:) =
[min(handles.planC{indexS.structures}(1,structNum(1)).contour(i).segments(1).points(:,3))
max(handles.planC{indexS.structures}(1,structNum(1)).contour(i).segments(1).points(:,3))];
    count = count + 1;
end

end

end
if ~isempty(handles.planC{indexS.scan}(1,1).transM)
    Lower = [0 0 min(x(:,1)) min(y(:,1)) min(z(:,1)) 0.0];
    Upper = [1 1 max(x(:,2)) max(y(:,2)) max(z(:,2)) 1.4];
    Lower(3) = Lower(3) + handles.planC{1,3}(1,1).transM(1,4);
    Upper(3) = Upper(3) + handles.planC{1,3}(1,1).transM(1,4);
    CalcUp4 = handles.planC{1,3}(1,1).transM(2,4) - Lower(4);
    CalcLow4 = handles.planC{1,3}(1,1).transM(2,4) - Upper(4);
    Lower(4) = CalcLow4;
    Upper(4) = CalcUp4;
    Lower(5) = Lower(5) + handles.planC{1,3}(1,1).transM(3,4);
    Upper(5) = Upper(5) + handles.planC{1,3}(1,1).transM(3,4);
    LowUp = [Lower;Upper];
else
    Lower = [0 0 min(x(:,1)) min(y(:,1)) min(z(:,1)) 0.0];
    Upper = [1 1 max(x(:,2)) max(y(:,2)) max(z(:,2)) 1.4];
    LowUp = [Lower;Upper]
end

%Setup and run the Genetic Algorithm
nvarsS = get(handles.edit3,'String');
nvars = str2num(nvarsS);
PopInitRange_Data = repmat(LowUp,1,nvars);
PopulationSize_DataS = get(handles.edit5,'String');
PopulationSize_Data = str2num(PopulationSize_DataS);
Generations_DataS = get(handles.edit4,'String');
Generations_Data = str2num(Generations_DataS);
%StallGenLimit_Data = round(0.5*Generations_Data);
lb = repmat(LowUp(1,:),1,nvars);
ub = repmat(LowUp(2,:),1,nvars);
nvars = nvars * 6;
PlotInterval = 1;

% Start with the default options
options = gaoptimset;
% Modify options setting
options = gaoptimset(options,'PopulationType','custom');
options = gaoptimset(options,'PopInitRange',PopInitRange_Data);
options = gaoptimset(options,'PopulationSize',PopulationSize_Data);
options = gaoptimset(options,'Generations',Generations_Data);

```

```

options = gaoptimset(options, 'StallGenLimit',
str2num(get(handles.edit10, 'String')));
options = gaoptimset(options, 'CreationFcn', @gacreationuniformTOGA);
options = gaoptimset(options, 'CrossoverFcn', @crossoverscatteredTOGA);
mutationcell = {@mutationgaussianTOGA
str2num(get(handles.edit8, 'String'))
str2num(get(handles.edit11, 'String'))};
options = gaoptimset(options, 'MutationFcn', mutationcell);
FitScale = get(handles.popupmenu4, 'Value');
%options = gaoptimset('UseParallel', 'always', 'Vectorized', 'off');
switch FitScale
    case 1
        options = gaoptimset(options, 'FitnessScalingFcn',
@fitscalingrank);
    case 2
        options = gaoptimset(options, 'FitnessScalingFcn',
@fitscalingprop);
end
SelFunc = get(handles.popupmenu5, 'Value');
switch SelFunc
    case 1
        options = gaoptimset(options, 'SelectionFcn',
@selectionstochunif);
    case 2
        options = gaoptimset(options, 'SelectionFcn',
@selectionremainder);
    case 3
        options = gaoptimset(options, 'SelectionFcn', @selectionuniform);
    case 4
        options = gaoptimset(options, 'SelectionFcn', @selectionroulette);
    case 5
        options = gaoptimset(options, 'SelectionFcn',
@selectiontournament);
end
options = gaoptimset(options, 'EliteCount',
str2num(get(handles.edit7, 'String')));
options = gaoptimset(options, 'CrossoverFraction',
str2num(get(handles.edit6, 'String')));
options = gaoptimset(options, 'Display', 'off');
options = gaoptimset(options, 'PlotInterval', PlotInterval);
options = gaoptimset(options, 'PlotFcns', { @gaplotbestf @gaplotdistance
@(options, state, flag)GAPlotDVHOpt(options, state, flag,
handles.planC, doseindex, Profiles, CalcMethod, gamma,
structNum, RxIsodose) @(options, state,
flag)GAPlotConformIndexOpt(options, state, flag, handles.planC,
doseindex, Profiles, CalcMethod, gamma, structNum, RxIsodose, TV) });
options = gaoptimset(options, 'UseParallel', 'always');

%Provide initial population if user provides
%initpopYN = get(handles.checkbox, 'Value');
initpopYN = 1;
if isempty(handles.planC{indexS.comment}).GAOptInitPopulation)
    fprintf(1, 'User selected initial population provided but an intial
population was not found.\n');

```

```

        fprintf(1, 'Optimization will be performed without an initial
population provided.\n\n');
    else
        X0 = handles.planC{indexS.comment}.GAOptInitPopulation;
        X0 = cell2mat(X0);
        options = gaoptimset(options, 'InitialPopulation', X0);
    end

    %Calculate scores for initial population
    for i = 1:size(X0,1)
        InitScore(i) =
        TOGAGAOptFunc(X0(i,:), handles.planC, doseindex, Profiles, gamma, w, structNu
m, RxIsodose, TV, CalcMethod);
    end
    options = gaoptimset(options, 'InitialScores', InitScore);

    %Run the actual Genetic Algorithm Optimization
    tic;
    [x, fval, exitflag, output, population, score] = ...
    ga(@ (para) TOGAGAOptFunc(para, handles.planC, doseindex, Profiles, gamma, w, s
tructNum, RxIsodose, TV, CalcMethod), nvars, [], [], [], [], lb, ub, [], options);
    tElapsed=toc;
    Message = ['GA optimization completed in ', num2str(tElapsed), '
sec',,];
    h = msgbox(Message, 'GA Optimization');

    %Run the hybrid optimization if user desires
    RunHybrid = get(handles.checkbox3, 'Value');
    if RunHybrid == 1
        numshot = size(x,2)/6;
        countc = 1;
        HybridCol = [];
        counts = 3;
        HybridStart = [];
        for i = 1:numshot
            HybridCol = [HybridCol x(1,[countc countc+1])];
            HybridStart = [HybridStart x(1,[counts counts+1 counts+2
counts+3])];
            countc = countc + 6;
            counts = counts + 6;
        end

        OptFuncHybrid =
        @(para) TOGAGAOptFuncHybrid(para, handles.planC, doseindex, Profiles, gamma,
w, structNum, RxIsodose, TV, CalcMethod, HybridCol);
        tic;
        [xhybrid, fvalhybrid] = fminunc(OptFuncHybrid, HybridStart);
        tElapsed=toc;
        HybridFReduce = fval - fvalhybrid;
        Message = ['Hybrid optimization completed in ', num2str(tElapsed),
' sec with a reduction of cost function value of: ',
num2str(HybridFReduce)];
        h = msgbox(Message, 'Hybrid Optimization');
        countx = 3;
        counthx = 1;
        for i = 1:numshot

```

```

        x(1,countx:countx+3) = xhybrid(1,counthx:counthx+3);
        countx = countx + 6;
        counthx = counthx + 4;
    end
end

%Import optimized shots into planC
para = x;
para = reshape(para,6,length(para)/6)';
numshots = size(para);
shots = zeros(numshots(1),6);
for i = 1:numshots(1)
    if isequal(para(i,1:2),[0 0])
        unbincol(i) = 4;
    elseif isequal(para(i,1:2),[0 1])
        unbincol(i) = 8;
    elseif isequal(para(i,1:2),[1 0])
        unbincol(i) = 14;
    elseif isequal(para(i,1:2),[1 1])
        unbincol(i) = 18;
    else
        error
    end
end
shots(:,1) = unbincol';
shots(:,2) = para(:,3);
shots(:,3) = para(:,4);
shots(:,4) = para(:,5);
shots(:,5) = para(:,6);
shots(:,6) = gamma;
shots(:,7) = 0;
shots = sortrows(shots,1)

count = 1;
for i = 1:numshots
    if shots(i,5) >= 0.09;
        shothold(1,count).collimator_helmet = shots(i,1);
        shothold(1,count).x = shots(i,2);
        shothold(1,count).y = shots(i,3);
        shothold(1,count).z = shots(i,4);
        %if shots(count,5) <= 0.05;
        %    shothold(1,count).weight = 0;
        %else
            shothold(1,count).weight = shots(i,5);
        %end
        shothold(1,count).gammaLeksell = shots(i,6);
        shothold(1,count).gammaTOGA = shots(i,6);
        shothold(1,count).fractionGroupID =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).fractionGroupID;
        shothold(1,count).associatedMatrix =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).associatedMatrix;
        shothold(1,count).targetpointLeksell =
[shots(i,2);shots(i,3);shots(i,4)];
    end
end

```

```

%handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).targetpoint
Leksell;
    shothold(1,count).targetpointTOGA =
[shots(i,2);shots(i,3);shots(i,4)];
%handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).targetpoint
TOGA;
    shothold(1,count).plugging =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).plugging;
    shothold(1,count).srcWeight =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).srcWeight;
    shothold(1,count).xyzSrcPositions =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).xyzSrcPositi
ons;
    shothold(1,count).skull =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).skull;
    shothold(1,count).time =
handles.planC{indexS.beamGeometry}(1,doseindex).shots(1,1).time;
    count = count + 1;
end
end

%Input optimized shots to planC
handles.planC{indexS.beamGeometry}(1,doseindex).shots = shothold;

%Input the results of the optimization into the comment section of
planC
planC{indexS.comment}.GAOptPopulation = population;

%Perform final dose calculation of optimzed plan
CalcMethod = 1;
[Profiles] = loadprofilesTOGA(handles.planC,doseindex);
[handles.planC] =
dosecalcTOGA(handles.planC,doseindex,Profiles,CalcMethod);

%Update planC with the new dose matrix position
assignin('base','planC',handles.planC);
guidata(hObject,handles);

% --- Executes on selection change in popupmenuGamma.
function popupmenuGamma_Callback(hObject,eventdata,handles)
% hObject      handle to popupmenuGamma (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenuGamma contents as cell array
%          contents{get(hObject,'Value')} returns selected item from
popupmenuGamma

% --- Executes during object creation, after setting all properties.
function popupmenuGamma_CreateFcn(hObject,eventdata,handles)
% hObject      handle to popupmenuGamma (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
% str2double(get(hObject,'String')) returns contents of edit3 as
a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject handle to edit4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
% str2double(get(hObject,'String')) returns contents of edit4 as
a double

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB

```



```

% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject      handle to edit5 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%      str2double(get(hObject,'String')) returns contents of edit5 as
a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit5 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function setGUIpopupmenuPlan(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)
    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
end
set(handles.popupmenuPlan, 'String', plannames);

function setGUIpopupmenuROI(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

```

```

%Get the ROIs names
ROIsize = size(handles.planC{handles.indexS.structures});
for i = 1:ROIsize(2)
    ROInames{i,1} =
handles.planC{handles.indexS.structures}(1,i).structureName;
end
set(handles.popupmenuStructure, 'String', ROInames);

function setGUIpopupmenuRingROI(handles)
% handles    structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the ROIs names
ROIsize = size(handles.planC{handles.indexS.structures});
for i = 1:ROIsize(2)
    ROInames{i,1} =
handles.planC{handles.indexS.structures}(1,i).structureName;
end
set(handles.popupmenu7, 'String', ROInames);

function setGUIpopupmenuCritROI1(handles)
% handles    structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the ROIs names
ROIsize = size(handles.planC{handles.indexS.structures});
for i = 1:ROIsize(2)
    ROInames{i,1} =
handles.planC{handles.indexS.structures}(1,i).structureName;
end
set(handles.popupmenu8, 'String', ROInames);

function setGUILabelSlideValue(handles)
% handles    structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

slidervalue = get(handles.slider1, 'Value');
slidervalueText = num2str(slidervalue);
set(handles.text1, 'String', slidervalueText);

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit6 as text
%         str2double(get(hObject, 'String')) returns contents of edit6 as
a double

```

```

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%         str2double(get(hObject,'String')) returns contents of edit7 as
a double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%         str2double(get(hObject,'String')) returns contents of edit8 as
a double

```

```

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in edit9.
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns edit9
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
edit9

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu4
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenu4

```

```

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu5.
function popupmenu5_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu5
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenu5

% --- Executes during object creation, after setting all properties.
function popupmenu5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%         str2double(get(hObject,'String')) returns contents of edit10
as a double

% --- Executes during object creation, after setting all properties.

```

```

function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%         str2double(get(hObject,'String')) returns contents of edit11
as a double

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

checkvalue = get(handles.checkbox1,'Value');
if checkvalue == 1
    set(handles.text19,'visible','on');
    set(handles.popupmenu7,'visible','on');
    setGUIDIpopupmenuRingROI(handles);
else

```

```

        set(handles.text19,'visible','off');
        set(handles.popupmenu7,'visible','off');
    end

% Update handles structure
guidata(hObject, handles);

% --- Executes on selection change in popupmenu7.
function popupmenu7_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu7
%          contents as cell array
%          contents{get(hObject,'Value')} returns selected item from
%          popupmenu7

% --- Executes during object creation, after setting all properties.
function popupmenu7_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
%              called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2

checkvalue = get(handles.checkbox2,'Value');
if checkvalue == 1
    set(handles.text20,'visible','on');
    set(handles.popupmenu8,'visible','on');
    set(handles.edit12,'visible','on');
    setGUIpopupmenuCritROI1(handles);
else
    set(handles.text20,'visible','off');
    set(handles.popupmenu8,'visible','off');
    set(handles.edit12,'visible','off');
end

% Update handles structure

```

```

guidata(hObject, handles);

% --- Executes on selection change in popupmenu8.
function popupmenu8_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu8 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu8
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu8

% --- Executes during object creation, after setting all properties.
function popupmenu8_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu8 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject      handle to edit12 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%           str2double(get(hObject,'String')) returns contents of edit12
as a double

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit12 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```



```
% --- Executes on button press in checkbox3.  
function checkbox3_Callback(hObject, eventdata, handles)  
% hObject      handle to checkbox3 (see GCBO)  
% eventdata    reserved - to be defined in a future version of MATLAB  
% handles      structure with handles and user data (see GUIDATA)  
  
% Hint: get(hObject,'Value') returns toggle state of checkbox3
```

PlanGUI

```
function varargout = PlanGUI(varargin)
% PLANGUI M-file for PlanGUI.fig
%     PLANGUI, by itself, creates a new PLANGUI or raises the existing
%     singleton*.
%
%     H = PLANGUI returns the handle to a new PLANGUI or the handle to
%     the existing singleton*.
%
%     PLANGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in PLANGUI.M with the given input
arguments.
%
%     PLANGUI('Property','Value',...) creates a new PLANGUI or raises
the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before PlanGUI_OpeningFunction gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to PlanGUI_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PlanGUI

% Last Modified by GUIDE v2.5 20-Dec-2010 21:16:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @PlanGUI_OpeningFcn, ...
                  'gui_OutputFcn',  @PlanGUI_OutputFcn, ...
                  'gui_LayoutFcn',  @PlanGUI_LayoutFcn, ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```

% --- Executes just before PlanGUI is made visible.
function PlanGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to PlanGUI (see VARARGIN)

% Choose default command line output for PlanGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
    %Populate objects in the GUI
    setGUIpopupmenuPlan(handles);
    seteditPlan(handles);
end

guidata(hObject, handles);

% UIWAIT makes PlanGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = PlanGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenuPlan.
function popupmenuPlan_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuPlan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

```

```

% Hints: contents = get(hObject,'String') returns popupmenuPlan
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
popupmenuPlan

% --- Executes during object creation, after setting all properties.
function popupmenuPlan_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuPlan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%         str2double(get(hObject,'String')) returns contents of edit1 as
a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text

```

```
%          str2double(get(hObject,'String')) returns contents of edit2 as
a double
```

```
% --- Executes during object creation, after setting all properties.
```

```
function edit2_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
```

```
%          See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
```

```
end
```

```
function edit3_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit3 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit3 as text
```

```
%          str2double(get(hObject,'String')) returns contents of edit3 as
a double
```

```
% --- Executes during object creation, after setting all properties.
```

```
function edit3_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit3 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
```

```
%          See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
```

```
end
```

```
function edit4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit4 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit4 as text
```

```
%          str2double(get(hObject,'String')) returns contents of edit4 as
a double
```

```

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%       str2double(get(hObject,'String')) returns contents of edit5 as
a double

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%       str2double(get(hObject,'String')) returns contents of edit6 as
a double

```

```

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%         str2double(get(hObject,'String')) returns contents of edit7 as
a double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

checkval = get(handles.checkbox1,'Value');

if checkval == 1
    set(handles.edit4,'Visible','off');
    set(handles.edit5,'Visible','off');

```

```

set(handles.edit6,'Visible','off');
set(handles.text8,'Visible','off');
set(handles.text9,'Visible','off');
set(handles.text10,'Visible','off');
set(handles.text14,'Visible','off');
set(handles.text15,'Visible','off');

Rx = str2num(get(handles.edit1,'String'));
set(handles.edit4,'String',num2str(Rx));
TxDate = get(handles.edit2,'String');
set(handles.edit5,'String',TxDate);
TxDR = str2num(get(handles.edit3,'String'));
set(handles.edit6,'String',num2str(TxDR));
else
    set(handles.edit4,'Visible','on');
    set(handles.edit5,'Visible','on');
    set(handles.edit6,'Visible','on');
    set(handles.text8,'Visible','on');
    set(handles.text9,'Visible','on');
    set(handles.text10,'Visible','on');
    set(handles.text14,'Visible','on');
    set(handles.text15,'Visible','on');
end

guidata(hObject, handles);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Get the selected method
if get(handles.radiobutton1,'Value') == 1
    method = 1;
elseif get(handles.radiobutton2,'Value') == 1
    method = 2;
elseif get(handles.radiobutton3,'Value') == 1
    method = 3;
end

%Get the plan data
planval = get(handles.popupmenuPlan,'Value');
%Rx = str2num(get(handles.edit1,'String'))
%TxDate = get(handles.edit2,'String')
%TxDR = str2num(get(handles.edit3,'String'))
MatrixName = get(handles.edit7,'String');
Rx = str2num(get(handles.edit4,'String'));
TxDate = get(handles.edit5,'String');
TxDR = str2num(get(handles.edit6,'String'));
copyplan = get(handles.checkbox1,'Value');

%Executes newdosegrid function
[handles.planC] =
newdosegridTOGA(handles.planC,method,MatrixName,Rx,TxDate,TxDR,copyplan
,planval);

```



```

%Update planC with the new plan data
assignin('base','planC',handles.planC);

guidata(hObject, handles);

function setGUIpopupmenuPlan(handles)
% handles    structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)
    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
    %plannames{i,1} = [handles.shots(i,1).fractionGroupID];
end
set(handles.popupmenuPlan,'String',plannames);

function seteditPlan(handles)
% handles    structure with handles and user data (see GUIDATA)
%Set editbox text based on listBoxShots selection
%
%Sets the text string of the edit boxes according to the value of
%handles.shots

popval = get(handles.popupmenuPlan,'Value');

Rx = handles.planC{handles.indexS.dose}(popval).Rx;
set(handles.edit1,'String',num2str(Rx));
DR = handles.planC{handles.indexS.dose}(popval).doserate;
set(handles.edit3,'String',num2str(DR));
TxDate = handles.planC{handles.indexS.dose}(popval).txdate;
set(handles.edit2,'String',TxDate);
numshots =
length(handles.planC{handles.indexS.beamGeometry}(popval).shots);
set(handles.txt,'String',numshots);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get the plan data
popval = get(handles.popupmenuPlan,'Value');
Rx = str2num(get(handles.edit1,'String'));
TxDate = get(handles.edit2,'String');
TxDR = str2num(get(handles.edit3,'String'));

```

```

%Update the handles.planC variable with the new data
indexS = handles.planC{end};
handles.planC{indexS.dose}(popval).Rx = Rx;
handles.planC{indexS.dose}(popval).txdate = TxDate;
handles.planC{indexS.dose}(popval).doserate = TxDR;

%Update planC with the new plan data
com = ['planC{', num2str(indexS.dose), '}'(' , num2str(popval), ').Rx = ', num2str(Rx) ';'];
evalin('base',com);
com = ['planC{', num2str(indexS.dose), '}'(' , num2str(popval), ').txdate = ', TxDate ' ';'];
evalin('base',com);
com = ['planC{', num2str(indexS.dose), '}'(' , num2str(popval), ').doserate = ', num2str(TxDR) ' ';'];
evalin('base',com);

guidata(hObject, handles);

% -----
function uipanel1_ButtonDownFcn(hObject, eventdata, handles)
% hObject      handle to uipanel1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%if get(handles.radiobutton1,'Value') == 1
%    a = 1
%elseif get(handles.radiobutton2,'Value') == 1
%    set(handles.checkbox1,'value',1);
%elseif get(handles.radiobutton3,'Value') == 1
%    set(handles.checkbox1,'value',1);
%end

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to uipanel1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% --- Creates and returns a handle to the GUI figure.
function h1 = PlanGUI_LayoutFcn(policy)
% policy - create a new figure or use a singleton. 'new' or 'reuse'.

persistent hsingleton;
if strcmpi(policy, 'reuse') & ishandle(hsingleton)
    h1 = hsingleton;
    return;
end
load PlanGUI.mat

```

```

appdata = [];
appdata.GUIDEOptions = struct(...
    'active_h', [], ...
    'taginfo', struct(...
    'figure', 2, ...
    'popupmenu', 2, ...
    'edit', 8, ...
    'text', 16, ...
    'uipanel', 2, ...
    'radiobutton', 4, ...
    'checkbox', 2, ...
    'pushbutton', 3), ...
    'override', 0, ...
    'release', 13, ...
    'resize', 'none', ...
    'accessibility', 'callback', ...
    'mfile', 1, ...
    'callbacks', 1, ...
    'singleton', 1, ...
    'syscolorfig', 1, ...
    'blocking', 0, ...
    'lastSavedFile', '/Users/jbowling/Documents/MATLAB/PlanGUI.m', ...
    'lastFilename', '/TOGA/CERR 3.3
Beta1/UserDefinedFunctions/TOGA/PlanGUI.fig');
appdata.lastValidTag = 'figure1';
appdata.GUIDELayoutEditor = [];
appdata.initTags = struct(...
    'handle', [], ...
    'tag', 'figure1');

h1 = figure(...
    'Units', 'characters', ...
    'Color', [0.701960784313725 0.701960784313725 0.701960784313725], ...
    'Colormap', [0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;0 0 0.8125;0 0
0.875;0 0 0.9375;0 0 1;0 0.0625 1;0 0.125 1;0 0.1875 1;0 0.25 1;0
0.3125 1;0 0.375 1;0 0.4375 1;0 0.5 1;0 0.5625 1;0 0.625 1;0 0.6875 1;0
0.75 1;0 0.8125 1;0 0.875 1;0 0.9375 1;0 1 1;0.0625 1 1;0.125 1
0.9375;0.1875 1 0.875;0.25 1 0.8125;0.3125 1 0.75;0.375 1 0.6875;0.4375
1 0.625;0.5 1 0.5625;0.5625 1 0.5;0.625 1 0.4375;0.6875 1 0.375;0.75 1
0.3125;0.8125 1 0.25;0.875 1 0.1875;0.9375 1 0.125;1 1 0.0625;1 1 0;1
0.9375 0;1 0.875 0;1 0.8125 0;1 0.75 0;1 0.6875 0;1 0.625 0;1 0.5625
0;1 0.5 0;1 0.4375 0;1 0.375 0;1 0.3125 0;1 0.25 0;1 0.1875 0;1 0.125
0;1 0.0625 0;1 0 0;0.9375 0 0;0.875 0 0;0.8125 0 0;0.75 0 0;0.6875 0
0;0.625 0 0;0.5625 0 0], ...
    'IntegerHandle', 'off', ...
    'InvertHardcopy', get(0, 'defaultfigureInvertHardcopy'), ...
    'MenuBar', 'none', ...
    'Name', 'PlanGUI', ...
    'NumberTitle', 'off', ...
    'PaperPosition', get(0, 'defaultfigurePaperPosition'), ...
    'Position', [103.714285714286 22.9285714285714 90.2857142857143
38.5714285714286], ...
    'Resize', 'off', ...
    'HandleVisibility', 'callback', ...

```

```

'Tag','figure1',...
'UserData',[],...
'Visible','on',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'popupmenuPlan';

h2 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback',@(hObject,eventdata)PlanGUI('popupmenuPlan_Callback',hObject
,eventdata,guidata(hObject)),...
'Position',[2.2 34.5384615384616 20 1.53846153846154],...
'String','Plan-',...
'Style','popupmenu',...
'Value',1,...
'CreateFcn',{@local_CreateFcn,
@(hObject,eventdata)PlanGUI('popupmenuPlan_CreateFcn',hObject,eventdata
,guidata(hObject)), appdata} ,...
'Tag','popupmenuPlan');

appdata = [];
appdata.lastValidTag = 'edit1';

h3 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback',@(hObject,eventdata)PlanGUI('edit1_Callback',hObject,eventda
ta,guidata(hObject)),...
'Position',[44.4 34.5384615384616 25.4 1.61538461538462],...
'String',blanks(0),...
'Style','edit',...
'CreateFcn',{@local_CreateFcn,
@(hObject,eventdata)PlanGUI('edit1_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
'Tag','edit1');

appdata = [];
appdata.lastValidTag = 'edit2';

h4 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback',@(hObject,eventdata)PlanGUI('edit2_Callback',hObject,eventda
ta,guidata(hObject)),...
'Position',[44.2 30.6923076923077 25.6 1.61538461538462],...
'String',blanks(0),...
'Style','edit',...
'CreateFcn',{@local_CreateFcn,
@(hObject,eventdata)PlanGUI('edit2_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
'Tag','edit2');

```

```

appdata = [];
appdata.lastValidTag = 'edit3';

h5 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'BackgroundColor',[1 1 1],...
    'Callback',@(hObject,eventdata)PlanGUI('edit3_Callback',hObject,eventda
ta,guidata(hObject)),...
    'Position',[44.2 26.8461538461538 25.8 1.61538461538462],...
    'String',blanks(0),...
    'Style','edit',...
    'CreateFcn',{@local_CreateFcn,
    @(hObject,eventdata)PlanGUI('edit3_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
    'Tag','edit3');

appdata = [];
appdata.lastValidTag = 'txt';

h6 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'Position',[51.2 23 10.6 1.15384615384615],...
    'String',{ 'Static Text' },...
    'Style','text',...
    'Tag','txt',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'textRx';

h7 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'Position',[29.8 34.7692307692308 10.6 1.15384615384615],...
    'String','Rx Dose',...
    'Style','text',...
    'Tag','textRx',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'textDate';

h8 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'Position',[29.8 30.7692307692308 10.6 1.15384615384615],...
    'String','Date',...
    'Style','text',...
    'Tag','textDate',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];

```

```

appdata.lastValidTag = 'txtDR';

h9 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'Position',[29.8 26.8461538461538 10.6 1.15384615384615],...
    'String','Dose Rate',...
    'Style','text',...
    'Tag','txtDR',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'txtShots';

h10 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'Position',[29.8 23 10.6 1.15384615384615],...
    'String','# Shots',...
    'Style','text',...
    'Tag','txtShots',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'txtPlan';

h11 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'HorizontalAlignment','left',...
    'Position',[2.6 36.5384615384615 10.6 1.15384615384615],...
    'String','Plans',...
    'Style','text',...
    'Tag','txtPlan',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'txtAddDosePlan';

h12 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'HorizontalAlignment','left',...
    'Position',[2.6 19.2307692307692 20.2 1.15384615384615],...
    'String','Add New Dose Plan',...
    'Style','text',...
    'Tag','txtAddDosePlan',...
    'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'uipanel1';

h13 = uibuttongroup(...
    'Parent',h1,...
    'Units','characters',...

```

```

'Title','New Dose Grid Method',...
'ButtonDownFcn',@(hObject,eventdata)PlanGUI('uipanel1_ButtonDownFcn',hO
bject,eventdata,guidata(hObject)),...
'CreateFcn',{@local_CreateFcn,
@(hObject,eventdata)PlanGUI('uipanel1_CreateFcn',hObject,eventdata,guid
ata(hObject)), appdata} ,...
'Tag','uipanel1',...
'Clipping','on',...
'Position',[1.2 8.07692307692307 27 10.5384615384615],...
'SelectedObject',[],...
'SelectionChangeFcn',@(hObject,eventdata)PlanGUI('uipanel1_SelectionCha
ngeFcn',get(hObject,'SelectedObject'),eventdata,guidata(get(hObject,'Se
lectedObject'))),...
'OldSelectedObject',[]);

appdata = [];
appdata.lastValidTag = 'radiobutton1';

h14 = uicontrol(...
'Parent',h13,...
'Units','characters',...
'Callback',mat{1},...
'Position',[4 7.15384615384616 16.4 1.15384615384615],...
'String','Graphical',...
'Style','radiobutton',...
'Value',1,...
'Tag','radiobutton1',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'radiobutton2';

h15 = uicontrol(...
'Parent',h13,...
'Units','characters',...
'Callback',mat{2},...
'Position',[4 4.30769230769231 16.4 1.15384615384615],...
'String','Shots',...
'Style','radiobutton',...
'Tag','radiobutton2',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'radiobutton3';

h16 = uicontrol(...
'Parent',h13,...
'Units','characters',...
'Callback',mat{3},...
'Position',[4 1.38461538461539 16.4 1.15384615384615],...
'String','Replicate',...
'Style','radiobutton',...
'Tag','radiobutton3',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

```

```

appdata = [];
appdata.lastValidTag = 'edit4';

h17 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'BackgroundColor',[1 1 1],...
    'Callback',@(hObject,eventdata)PlanGUI('edit4_Callback',hObject,eventda
ta,guidata(hObject)),...
    'Position',[44.4 13.7692307692308 25.4 1.61538461538462],...
    'String',blanks(0),...
    'Style','edit',...
    'CreateFcn',{@local_CreateFcn,
    @(hObject,eventdata)PlanGUI('edit4_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
    'Tag','edit4');

appdata = [];
appdata.lastValidTag = 'edit5';

h18 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'BackgroundColor',[1 1 1],...
    'Callback',@(hObject,eventdata)PlanGUI('edit5_Callback',hObject,eventda
ta,guidata(hObject)),...
    'Position',[44.2 9.84615384615384 25.6 1.61538461538462],...
    'String',blanks(0),...
    'Style','edit',...
    'CreateFcn',{@local_CreateFcn,
    @(hObject,eventdata)PlanGUI('edit5_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
    'Tag','edit5');

appdata = [];
appdata.lastValidTag = 'edit6';

h19 = uicontrol(...
    'Parent',h1,...
    'Units','characters',...
    'BackgroundColor',[1 1 1],...
    'Callback',@(hObject,eventdata)PlanGUI('edit6_Callback',hObject,eventda
ta,guidata(hObject)),...
    'Position',[44.2 5.92307692307691 25.8 1.61538461538462],...
    'String',blanks(0),...
    'Style','edit',...
    'CreateFcn',{@local_CreateFcn,
    @(hObject,eventdata)PlanGUI('edit6_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
    'Tag','edit6');

appdata = [];
appdata.lastValidTag = 'text8';

h20 = uicontrol(...

```



```

'Parent',h1,...
'Units','characters',...
'Position',[29.8 14 10.6 1.15384615384615],...
'String','Rx Dose',...
'Style','text',...
'Tag','text8',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text9';

h21 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[29.8 9.92307692307691 10.6 1.15384615384615],...
'String','Date',...
'Style','text',...
'Tag','text9',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text10';

h22 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[29.8 5.92307692307691 10.6 1.15384615384615],...
'String','Dose Rate',...
'Style','text',...
'Tag','text10',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'edit7';

h23 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback',@(hObject,eventdata)PlanGUI('edit7_Callback',hObject,eventda
ta,guidata(hObject)),...
'Position',[44.4 17.5384615384615 25.4 1.61538461538462],...
'String',blanks(0),...
'Style','edit',...
'CreateFcn',{@local_CreateFcn,
@(hObject,eventdata)PlanGUI('edit7_CreateFcn',hObject,eventdata,guidata
(hObject)), appdata} ,...
'Tag','edit7');

appdata = [];
appdata.lastValidTag = 'text11';

h24 = uicontrol(...
'Parent',h1,...
'Units','characters',...

```

```

'Position',[29.8 17.7692307692308 10.6 1.15384615384615],...
'String','Grid Name',...
'Style','text',...
'Tag','text11',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'checkbox1';

h25 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback',@(hObject,eventdata)PlanGUI('checkbox1_Callback',hObject,eventdata,guidata(hObject)),...
'Position',[44 2.6153846153846 30.4 1.23076923076923],...
'String','Copy selected plan',...
'Style','checkbox',...
'Tag','checkbox1',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'pushbutton1';

h26 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback',@(hObject,eventdata)PlanGUI('pushbutton1_Callback',hObject,eventdata,guidata(hObject)),...
'Position',[2.2 1.99999999999999 22 1.76923076923077],...
'String','Add New Dose Grid',...
'Tag','pushbutton1',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text12';

h27 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[72.4 34.9230769230769 10.6 1.15384615384615],...
'String','Gy (max)',...
'Style','text',...
'Tag','text12',...
'CreateFcn',{@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text13';

h28 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[72.4 27 10.6 1.15384615384615],...
'String','Gy/min',...
'Style','text',...
'Tag','text13',...

```

```

'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text14';

h29 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[72.8 13.7692307692308 10.6 1.15384615384615],...
'String','Gy (max)',...
'Style','text',...
'Tag','text14',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'text15';

h30 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Position',[72.8 5.84615384615385 10.6 1.15384615384615],...
'String','Gy/min',...
'Style','text',...
'Tag','text15',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

appdata = [];
appdata.lastValidTag = 'pushbutton2';

h31 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'Callback',@(hObject,eventdata)PlanGUI('pushbutton2_Callback',hObject,e
ventdata,guidata(hObject)),...
'Position',[2.2 29 21.8 1.76923076923077],...
'String','Set Dose Changes',...
'Tag','pushbutton2',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );

hsingleton = h1;

% --- Set application data first then calling the CreateFcn.
function local_CreateFcn(hObject, eventdata, createfcn, appdata)

if ~isempty(appdata)
    names = fieldnames(appdata);
    for i=1:length(names)
        name = char(names(i));
        setappdata(hObject, name, getfield(appdata,name));
    end
end

```

```

if ~isempty(createfcn)
    if isa(createfcn, 'function_handle')
        createfcn(hObject, eventdata);
    else
        eval(createfcn);
    end
end

% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)

gui_StateFields = {'gui_Name'
    'gui_Singleton'
    'gui_OpeningFcn'
    'gui_OutputFcn'
    'gui_LayoutFcn'
    'gui_Callback'};
gui_Mfile = '';
for i=1:length(gui_StateFields)
    if ~isfield(gui_State, gui_StateFields{i})
        error('MATLAB:gui_mainfcn:FieldNotFound', 'Could not find field
%s in the gui_State struct in GUI M-file %s', gui_StateFields{i},
gui_Mfile);
    elseif isequal(gui_StateFields{i}, 'gui_Name')
        gui_Mfile = [gui_State.(gui_StateFields{i}), '.m'];
    end
end

numargin = length(varargin);

if numargin == 0
    % PLANGUI
    % create the GUI only if we are not in the process of loading it
    % already
    gui_Create = true;
elseif local_isInvokeActiveXCallback(gui_State, varargin{:})
    % PLANGUI(ACTIVEX,...)
    vin{1} = gui_State.gui_Name;
    vin{2} = [get(varargin{1}.Peer, 'Tag'), '_', varargin{end}];
    vin{3} = varargin{1};
    vin{4} = varargin{end-1};
    vin{5} = guidata(varargin{1}.Peer);
    feval(vin{:});
    return;
elseif local_isInvokeHGCallback(gui_State, varargin{:})
    % PLANGUI('CALLBACK',hObject,eventData,handles,...)
    gui_Create = false;
else
    % PLANGUI(...)
    % create the GUI and hand varargin to the openingfcn
    gui_Create = true;
end

if ~gui_Create

```

```

        % In design time, we need to mark all components possibly created
in
        % the coming callback evaluation as non-serializable. This way,
they
        % will not be brought into GUIDE and not be saved in the figure
file
        % when running/saving the GUI from GUIDE.
designEval = false;
if (numargin>1 && ishghandle(varargin{2}))
    fig = varargin{2};
    while ~isempty(fig) && ~isa(handle(fig), 'figure')
        fig = get(fig, 'parent');
    end

    designEval = isappdata(0, 'CreatingGUIDEFigure') ||
isprop(fig, '__GUIDEFigure');
end

if designEval
    beforeChildren = findall(fig);
end

% evaluate the callback now
varargin{1} = gui_State.gui_Callback;
if nargout
    [varargout{1:nargout}] = feval(varargin{:});
else
    feval(varargin{:});
end

% Set serializable of objects created in the above callback to off
in
% design time. Need to check whether figure handle is still valid
in
% case the figure is deleted during the callback dispatching.
if designEval && ishandle(fig)
    set(setdiff(findall(fig), beforeChildren),
'Serializable', 'off');
end
else
if gui_State.gui_Singleton
    gui_SingletonOpt = 'reuse';
else
    gui_SingletonOpt = 'new';
end

% Check user passing 'visible' P/V pair first so that its value can
be
% used by oepnfig to prevent flickering
gui_Visible = 'auto';
gui_VisibleInput = '';
for index=1:2:length(varargin)
    if length(varargin) == index || ~ischar(varargin{index})
        break;
    end
end

```

```

        % Recognize 'visible' P/V pair
        len1 = min(length('visible'),length(varargin{index}));
        len2 = min(length('off'),length(varargin{index+1}));
        if ischar(varargin{index+1}) &&
strncmpi(varargin{index},'visible',len1) && len2 > 1
            if strncmpi(varargin{index+1},'off',len2)
                gui_Visible = 'invisible';
                gui_VisibleInput = 'off';
            elseif strncmpi(varargin{index+1},'on',len2)
                gui_Visible = 'visible';
                gui_VisibleInput = 'on';
            end
        end
    end
end

% Open fig file with stored settings. Note: This executes all
component
% specific CreateFunctions with an empty HANDLES structure.

% Do feval on layout code in m-file if it exists
gui_Exported = ~isempty(gui_State.gui_LayoutFcn);
% this application data is used to indicate the running mode of a
GUIDE
% GUI to distinguish it from the design mode of the GUI in GUIDE.
it is
% only used by actxproxy at this time.
setappdata(0,genvarname(['OpenGuiWhenRunning_',
gui_State.gui_Name]),1);
if gui_Exported
    gui_hFigure = feval(gui_State.gui_LayoutFcn, gui_SingletonOpt);

% make figure invisible here so that the visibility of figure
is
% consistent in OpeningFcn in the exported GUI case
if isempty(gui_VisibleInput)
    gui_VisibleInput = get(gui_hFigure,'Visible');
end
set(gui_hFigure,'Visible','off')

% openfig (called by local_openfig below) does this for guis
without
% the LayoutFcn. Be sure to do it here so guis show up on
screen.
movegui(gui_hFigure,'onscreen');
else
    gui_hFigure = local_openfig(gui_State.gui_Name,
gui_SingletonOpt, gui_Visible);
% If the figure has InGUIInitialization it was not completely
created
% on the last pass. Delete this handle and try again.
if isappdata(gui_hFigure, 'InGUIInitialization')
    delete(gui_hFigure);
    gui_hFigure = local_openfig(gui_State.gui_Name,
gui_SingletonOpt, gui_Visible);
end
end

```

```

end
if isappdata(0, genvarname(['OpenGuiWhenRunning_',
gui_State.gui_Name]))
    rmapdata(0, genvarname(['OpenGuiWhenRunning_',
gui_State.gui_Name]));
end

% Set flag to indicate starting GUI initialization
setappdata(gui_hFigure, 'InGUIInitialization', 1);

% Fetch GUIDE Application options
gui_Options = getappdata(gui_hFigure, 'GUIDEOptions');
% Singleton setting in the GUI M-file takes priority if different
gui_Options.singleton = gui_State.gui_Singleton;

if ~isappdata(gui_hFigure, 'GUIOnScreen')
    % Adjust background color
    if gui_Options.syscolorfig
        set(gui_hFigure, 'Color',
get(0, 'DefaultUiControlBackgroundColor'));
    end

    % Generate HANDLES structure and store with GUIDATA. If there
is
    % user set GUI data already, keep that also.
    data = guidata(gui_hFigure);
    handles = guihandles(gui_hFigure);
    if ~isempty(handles)
        if isempty(data)
            data = handles;
        else
            names = fieldnames(handles);
            for k=1:length(names)
                data.(char(names(k)))=handles.(char(names(k)));
            end
        end
    end
    guidata(gui_hFigure, data);
end

% Apply input P/V pairs other than 'visible'
for index=1:2:length(varargin)
    if length(varargin) == index || ~ischar(varargin{index})
        break;
    end

    len1 = min(length('visible'), length(varargin{index}));
    if ~strncmpi(varargin{index}, 'visible', len1)
        try set(gui_hFigure, varargin{index}, varargin{index+1}),
catch break, end
    end
end

% If handle visibility is set to 'callback', turn it on until
finished

```

```

    % with OpeningFcn
    gui_HandleVisibility = get(gui_hFigure, 'HandleVisibility');
    if strcmp(gui_HandleVisibility, 'callback')
        set(gui_hFigure, 'HandleVisibility', 'on');
    end

    feval(gui_State.gui_OpeningFcn, gui_hFigure, [],
guidata(gui_hFigure), varargin{:});

    if isscalar(gui_hFigure) && ishandle(gui_hFigure)
        % Handle the default callbacks of predefined toolbar tools in
this
        % GUI, if any

%guidemfile('restoreToolbarToolPredefinedCallback',gui_hFigure);

        % Update handle visibility
        set(gui_hFigure, 'HandleVisibility', gui_HandleVisibility);

        % Call openfig again to pick up the saved visibility or apply
the
        % one passed in from the P/V pairs
        if ~gui_Exported
            gui_hFigure = local_openfig(gui_State.gui_Name,
'reuse',gui_Visible);
        elseif ~isempty(gui_VisibleInput)
            set(gui_hFigure, 'Visible', gui_VisibleInput);
        end
        if strcmpi(get(gui_hFigure, 'Visible'), 'on')
            figure(gui_hFigure);

            if gui_Options.singleton
                setappdata(gui_hFigure, 'GUIOnScreen', 1);
            end
        end

        % Done with GUI initialization
        if isappdata(gui_hFigure, 'InGUIInitialization')
            rmappdata(gui_hFigure, 'InGUIInitialization');
        end

        % If handle visibility is set to 'callback', turn it on until
        % finished with OutputFcn
        gui_HandleVisibility = get(gui_hFigure, 'HandleVisibility');
        if strcmp(gui_HandleVisibility, 'callback')
            set(gui_hFigure, 'HandleVisibility', 'on');
        end
        gui_Handles = guidata(gui_hFigure);
    else
        gui_Handles = [];
    end

    if narginout
        [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn,
gui_hFigure, [], gui_Handles);
    end

```



```

else
    feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
end

if isscalar(gui_hFigure) && ishandle(gui_hFigure)
    set(gui_hFigure, 'HandleVisibility', gui_HandleVisibility);
end
end

function gui_hFigure = local_openfig(name, singleton, visible)

% openfig with three arguments was new from R13. Try to call that
first, if
% failed, try the old openfig.
if nargin('openfig') == 2
    % OPENFIG did not accept 3rd input argument until R13,
    % toggle default figure visible to prevent the figure
    % from showing up too soon.
    gui_OldDefaultVisible = get(0, 'defaultFigureVisible');
    set(0, 'defaultFigureVisible', 'off');
    gui_hFigure = openfig(name, singleton);
    set(0, 'defaultFigureVisible', gui_OldDefaultVisible);
else
    gui_hFigure = openfig(name, singleton, visible);
end

function result = local_isInvokeActiveXCallback(gui_State, varargin)

try
    result = ispc && iscom(varargin{1}) ...
        && isequal(varargin{1}, gcbo);
catch
    result = false;
end

function result = local_isInvokeHGCallback(gui_State, varargin)

try
    fhandle = functions(gui_State.gui_Callback);
    result = ~isempty(findstr(gui_State.gui_Name, fhandle.file)) || ...
        (ischar(varargin{1}) ...
        && isequal(ishandle(varargin{2}), 1) ...
        && (~isempty(strfind(varargin{1}, [get(varargin{2}, 'Tag'),
        '_' ])) || ...
        ~isempty(strfind(varargin{1}, '_CreateFcn')));
catch
    result = false;
end

% --- Executes when selected object is changed in uipanel1.
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel1

```

```
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none
was selected
%   NewValue: handle of the currently selected object
% handles      structure with handles and user data (see GUIDATA)
```

PriorPlanSelect

```
function [SelectShots] = PriorPlanSelect(structNum, planC)

%Initialize the function
structNum = 1;
indexS = planC{end};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Calculate the comparison parameters for the selected structure
output = structureStats(1, planC);

%Get the strucute volume
TV = output.vol;

%Get the center of mass of the structure
COM = output.COM;

%Get the largest dimension in X, Y, Z of the structure
LargeDim = output.max - output.min;

%Get the contour points
numslices = length(planC{indexS.structures}(1,structNum).contour);
structPoints = [];
for i = 1:numslices;
    if
~isempty(planC{indexS.structures}(1,structNum(1)).contour(1,i).segments
.points)
        temp =
planC{indexS.structures}(1,structNum).contour(1,i).segments.points;
        tempsize = size(temp,1);
        strPsize = size(structPoints,1);
        structPoints(strPsize+1:strPsize+tempsize,:) = [temp
repmat(i,tempsize,1)];
    end
end
structPoints(:,3) = structPoints(:,3) +
planC{indexS.scan}(1,1).transM(3,4);

%Get the raster segments for the structure
[structRasterSegments] = getRasterSegments(structNum, planC);

%Calculate the spread of the structure from the center of mass
structDistCOM = (sum((structPoints(:,[1 2 3]) -
repmat(COM,size(structPoints,1),1)).^2,2)).^(1/2);
structDistMeanCOM = mean(structDistCOM);
structDistVarCOM = var(structDistCOM);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%PCA analysis
%Plot the original data
```

```

figure(10);plot3(structPoints(:,1),structPoints(:,2),structPoints(:,3),
'bo');
grid on;
%Perform the PCA
[coeff,score,roots,t2] = princomp(structPoints(:,1:3));
basis = coeff(:,1:2)
normal = coeff(:,3)
pctExplained = roots' ./ sum(roots)

[n,p] = size(structPoints(:,1:3));
meanstructPoints = mean(structPoints(:,1:3),1);
structPointsFit = repmat(meanstructPoints,n,1) +
score(:,1:2)*coeff(:,1:2)';
residuals = structPoints(:,1:3) - structPointsFit;

%Calculate PCA errors
error = abs((structPoints(:,1:3) -
repmat(meanstructPoints,n,1))*normal);
PCA_SSE = sum(error.^2); %SSE of the PCA
PCA_MeanError = mean(error); %Mean error of the PCA

%Calculate PCA Hotelling's T2
PCA_t2Mean = mean(t2);
PCA_t2Min = min(t2);
PCA_t2Max = max(t2);

%Plot the PCA
[xgrid,ygrid] =
meshgrid(linspace(min(structPoints(:,1)),max(structPoints(:,1)),5), ...
linspace(min(structPoints(:,2)),max(structPoints(:,2)),5));
zgrid = (1/normal(3)) .* (meanstructPoints*normal - (xgrid.*normal(1) +
ygrid.*normal(2)));
h = mesh(xgrid,ygrid,zgrid,'EdgeColor',[0 0 0],'FaceAlpha',0);
hold on
above = (structPoints(:,1:3)-repmat(meanstructPoints,n,1))*normal > 0;
below = ~above;
nabove = sum(above);
structPoints1 = [structPoints(above,1) structPointsFit(above,1)
nan*ones(nabove,1)];
structPoints2 = [structPoints(above,2) structPointsFit(above,2)
nan*ones(nabove,1)];
structPoints3 = [structPoints(above,3) structPointsFit(above,3)
nan*ones(nabove,1)];
plot3(structPoints1',structPoints2',structPoints3','- ',
structPoints(above,1),structPoints(above,2),structPoints(above,3),'o',
'Color',[0 .7 0]);
nbelow = sum(below);
structPoints1 = [structPoints(below,1) structPointsFit(below,1)
nan*ones(nbelow,1)];
structPoints2 = [structPoints(below,2) structPointsFit(below,2)
nan*ones(nbelow,1)];
structPoints3 = [structPoints(below,3) structPointsFit(below,3)
nan*ones(nbelow,1)];
plot3(structPoints1',structPoints2',structPoints3','- ',
structPoints(below,1),structPoints(below,2),structPoints(below,3),'o',
'Color',[1 0 0]);

```

```

hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
TV
LargeDim
structDistMeanCOM
structDistVarCOM
PCA_SSE
PCA_MeanError
PCA_t2Mean
PCA_t2Min
PCA_t2Max
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

SelectShots = [1];

```

RevisePlanC

```
function [planC] = RevisePlanC(planC);
%
%[planC] = RevisePlanC(planC);
%
%Adds a blank shot to each existing plan for Gamma Plan data imported
from
%the DICOM format.

indexS = planC{end};

%Load a blank shot
str = computer;
if strcmp(str, 'PCWIN')
    load('C:\TOGA\CERR 3.3
Beta1\UserDefinedFunctions\TOGA\planCShotBlank.mat');
elseif strcmp(str, 'MACI64')
    load('/TOGA/CERR 3.3
Beta1/UserDefinedFunctions/TOGA/planCShotBlank.mat');
end

%Determine the number of plans
numplans = length(planC{indexS.dose});

%Add the shot
for i = 1:numplans
    planC{indexS.beamGeometry}(i).shots = planCShotBlank;
end

%Add a unity dose rate to each plan
for i = 1:numplans
    planC{indexS.dose}(i).Rx =
max(max(max(planC{indexS.dose}(1,i).doseArray)));
    planC{indexS.dose}(i).doserate = 1;
    planC{indexS.dose}(i).txdate = date;
end

%Add the dimension of the z dose grid
for i = 1:numplans
    planC{indexS.dose}(i).depthGridInterval =
planC{indexS.dose}(i).horizontalGridInterval;
end

%Calculate the transformation for the Leksell coordinate system to the
%DICOM coordinate system
%Coordinates of the fiducials
[Fiducials] = ginput(6);
LeksellCord = [5 160;5 1000;5 40;195 160;195 1000;195 40]/10;
FidDiff = LeksellCord - Fiducials;
```

```

ZFidDist = [Fiducials(1,2)-Fiducials(2,2) Fiducials(2,2)-Fiducials(3,2)
Fiducials(4,2)-Fiducials(5,2) Fiducials(5,2)-Fiducials(6,2)];
ZTopDist = mean([ZFidDist(1) ZFidDist(3)]);
ZBottomDist = mean([ZFidDist(2) ZFidDist(4)]);
ZLeksellSlice = 16 - ZTopDist;
prompt = {'Enter the dicom slice Z value:'};dlg_title = 'DICOM Z
Value';num_lines = 1;def = {''};
answer = inputdlg(prompt,dlg_title,num_lines,def);
ZDicomSlice = cell2mat(answer);
ZDicomSlice = str2num(ZDicomSlice);
Check = 12 - (ZTopDist + ZBottomDist);%Should be 12

Transform = [mean(FidDiff(:,1));mean([FidDiff(1,2) FidDiff(3,2)
FidDiff(4,2) FidDiff(6,2)]);ZLeksellSlice - ZDicomSlice(1)];
planC{indexS.comment}(1).LeksellTransform = Transform;

```

ShotsGUI

```
function varargout = ShotsGUI(varargin)
% SHOTSGUI M-file for ShotsGUI.fig
%     SHOTSGUI, by itself, creates a new SHOTSGUI or raises the
existing
%     singleton*.
%
%     H = SHOTSGUI returns the handle to a new SHOTSGUI or the handle
to
%     the existing singleton*.
%
%     SHOTSGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in SHOTSGUI.M with the given input
arguments.
%
%     SHOTSGUI('Property','Value',...) creates a new SHOTSGUI or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before ShotsGUI_OpeningFunction gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to ShotsGUI_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ShotsGUI

% Last Modified by GUIDE v2.5 02-Dec-2011 11:26:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @ShotsGUI_OpeningFcn, ...
                  'gui_OutputFcn',  @ShotsGUI_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```



```

% End initialization code - DO NOT EDIT

% --- Executes just before ShotsGUI is made visible.
function ShotsGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to ShotsGUI (see VARARGIN)

% Choose default command line output for ShotsGUI
handles.output = hObject;

%Get the planC variable from the base workspace
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found. Please use File->Open Patient
to load file.\n\n\n\n')
else
    handles.indexS = handles.planC{end};
    for i = 1:length(handles.planC{handles.indexS.beamGeometry})
        handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
    end
    %Populate objects in the GUI
    setGUIpopupmenuPlan(handles);
    setGUIlistboxShots(handles);
    seteditXYZ(handles);
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ShotsGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ShotsGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function editY_Callback(hObject, eventdata, handles)
% hObject    handle to editY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

```

```
% Hints: get(hObject,'String') returns contents of editY as text
%         str2double(get(hObject,'String')) returns contents of editY as
a double
```

```
% --- Executes during object creation, after setting all properties.
function editY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editZ_Callback(hObject, eventdata, handles)
% hObject    handle to editZ (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of editZ as text
%         str2double(get(hObject,'String')) returns contents of editZ as
a double
```

```
% --- Executes during object creation, after setting all properties.
function editZ_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editZ (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editWeight_Callback(hObject, eventdata, handles)
% hObject    handle to editWeight (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of editWeight as text
%         str2double(get(hObject,'String')) returns contents of
editWeight as a double
```

```

% --- Executes during object creation, after setting all properties.
function editWeight_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editWeight (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editGamma_Callback(hObject, eventdata, handles)
% hObject    handle to editGamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editGamma as text
%       str2double(get(hObject,'String')) returns contents of
editGamma as a double

% --- Executes during object creation, after setting all properties.
function editGamma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editGamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbuttonSet.
function pushbuttonSet_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonSet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

setnewshot(hObject, eventdata, handles);
%plotshotsifchecked(handles);

% --- Executes on selection change in popupmenuPlan.
function popupmenuPlan_Callback(hObject, eventdata, handles)

```

```

% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenuPlan
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuPlan

setGUIlistboxShots(handles);

% --- Executes during object creation, after setting all properties.
function popupmenuPlan_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuPlan (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in listBoxShots.
function listBoxShots_Callback(hObject, eventdata, handles)
% hObject      handle to listBoxShots (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listBoxShots contents
as cell array
%           contents{get(hObject,'Value')} returns selected item from
listBoxShots

seteditXYZ(handles)

% --- Executes during object creation, after setting all properties.
function listBoxShots_CreateFcn(hObject, eventdata, handles)
% hObject      handle to listBoxShots (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: listBox controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu2

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuHelmet.
function popupmenuHelmet_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenuHelmet (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenuHelmet
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenuHelmet

% --- Executes during object creation, after setting all properties.
function popupmenuHelmet_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenuHelmet (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function update_listbox(handles)
% Updates the shot listbox to selected plan
set(handles.listboxShots, 'String', vars)

function editX_Callback(hObject, eventdata, handles)
% hObject      handle to editX (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of editX as text
%         str2double(get(hObject, 'String')) returns contents of editX as
%         a double

% --- Executes during object creation, after setting all properties.
function editX_CreateFcn(hObject, eventdata, handles)
% hObject      handle to editX (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
%              called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function setGUIpopupmenuPlan(handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Populates the popupmenu

%Get the plan names
shotsize = size(handles.shots);
for i = 1:shotsize(2)
    plannames{i,1} = [handles.shots{i}(1).fractionGroupID];
end
set(handles.popupmenuPlan, 'String', plannames);

function setGUIlistboxShots(handles)
% handles      structure with handles and user data (see GUIDATA)
%Set listboxShots text
%
%Populates the listbox

%Get the shot names
shotsize = size(handles.shots);

```

```

numplans = shotsize(2);
for i = 1:numplans
    numshotsinplans(i) = length(handles.shots{i});
end
for i = 1:numplans
    for j = 1:numshotsinplans(i)
        shotname{j,i} = [handles.shots{i}(1,1).fractionGroupID(1:1) '-'
num2str(j)]; %Working Here 4-15-2011
        %shotname{j,i} = [handles.shots{j,i}.fractionGroupID(1:1) '-'
num2str(i)];
    end
end
val = get(handles.popupmenuPlan, 'Value');
set(handles.listboxShots, 'String', shotname(:, val));

function seteditXYZ(handles)
% handles      structure with handles and user data (see GUIDATA)
%Set editbox text based on listboxShots selection
%
%Sets the text string of the edit boxes according to the value of
%handles.shots

indexS = handles.planC{end};

popval = get(handles.popupmenuPlan, 'Value');
listval = get(handles.listboxShots, 'Value');

x = handles.shots{popval}(listval).x;
y = handles.shots{popval}(listval).y;
z = handles.shots{popval}(listval).z;
weight = handles.shots{popval}(listval).weight;
time = handles.shots{popval}(listval).time;
gamma = handles.shots{popval}(listval).gammaTOGA;
helmet = handles.shots{popval}(listval).collimator_helmet;

set(handles.editX, 'String', num2str(x));
set(handles.editY, 'String', num2str(y));
set(handles.editZ, 'String', num2str(z));
set(handles.editWeight, 'String', num2str(weight));
set(handles.editTime, 'String', num2str(time));
set(handles.editGamma, 'String', num2str(gamma));
switch helmet
case 4
    set(handles.popupmenuHelmet, 'Value', 1);
case 8
    set(handles.popupmenuHelmet, 'Value', 2);
case 14
    set(handles.popupmenuHelmet, 'Value', 3);
case 18
    set(handles.popupmenuHelmet, 'Value', 4);
end

checkval = get(handles.checkbox3, 'Value');
if checkval == 1
    Transform = handles.planC{indexS.comment}.LeksellTransform;
    xt = x + Transform(1);

```

```

        yt = y + Transform(2);
        zt = z + Transform(3);
        set(handles.textXorg, 'String', num2str(xt));
        set(handles.textYorg, 'String', num2str(yt));
        set(handles.textZorg, 'String', num2str(zt));
    else
        set(handles.textXorg, 'String', num2str(x));
        set(handles.textYorg, 'String', num2str(y));
        set(handles.textZorg, 'String', num2str(z));
    end

    set(handles.textWeightorg, 'String', num2str(weight));
    set(handles.textTimeorg, 'String', num2str(time));
    set(handles.textGammaorg, 'String', num2str(gamma));
    set(handles.textHelmetorg, 'String', num2str(helmet));

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject, 'Value') returns toggle state of checkbox1

if get(handles.checkbox1, 'Value');
    plotshotsifchecked(handles);
end

% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject, 'Value') returns toggle state of checkbox2

if get(handles.checkbox2, 'Value');
    % plotshotsifchecked(handles);
end

function plotshotsifchecked(handles)
% handles      structure with handles and user data (see GUIDATA)
% Sets the popupmenuPlan selection choices
%
% Calls plot3DSphere function if checkboxes selected

planindex = get(handles.popupmenuPlan, 'Value');
plotsurf = get(handles.checkbox2, 'Value');
plotpoints = get(handles.checkbox1, 'Value');

plot3DSphere(handles.shots, planindex, plotsurf, plotpoints);

```



```

function setnewshot(hObject, eventdata, handles)
% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Function that saves shots position to handles.shots and planC

indexS = handles.planC{end};

%Get the currently selected shot
planindex = get(handles.popupmenuPlan, 'Value');
shotindex = get(handles.listboxShots, 'Value');

%Get the currently selected shots new info
x = str2num(get(handles.editX, 'String'));
y = str2num(get(handles.editY, 'String'));
z = str2num(get(handles.editZ, 'String'));

checkval = get(handles.checkbox3, 'Value');
if checkval == 1
    %if exist(handles.planC{indexS.comment}.LeksellTransform)
    Transform = handles.planC{indexS.comment}.LeksellTransform;
    x = x - Transform(1);
    y = y - Transform(2);
    z = z - Transform(3);
    %end
end

Weight = str2num(get(handles.editWeight, 'String'));
Time = str2num(get(handles.editTime, 'String'));
Gamma = str2num(get(handles.editGamma, 'String'));
helmetval = get(handles.popupmenuHelmet, 'Value');
helmetstring_list = get(handles.popupmenuHelmet, 'String');
helmet = helmetstring_list{helmetval};

%Set the shots variable with the new position
handles.shots{planindex}(shotindex).x = x;
handles.shots{planindex}(shotindex).y = y;
handles.shots{planindex}(shotindex).z = z;
handles.shots{planindex}(shotindex).weight = Weight;
handles.shots{planindex}(shotindex).time = Time;
handles.shots{planindex}(shotindex).gammaTOGA = Gamma;
handles.shots{planindex}(shotindex).gammaLeksell = Gamma;
handles.shots{planindex}(shotindex).collimator_helmet = helmet;

%Update planC with the new shot position
indexS = handles.planC{end};
variable = ['planC{', num2str(indexS.beamGeometry), '} (1, ',
num2str(planindex), ').shots (1, ', num2str(shotindex), ') '];
variablex = [variable, '.x = ', num2str(x), ';'];
evalin('base', variablex);
variabley = [variable, '.y = ', num2str(y), ';'];
evalin('base', variabley);
variablez = [variable, '.z = ', num2str(z), ';'];

```

```

evalin('base',variablez);
variableWeight = [variable, '.weight = ', num2str(Weight), ';'];
evalin('base',variableWeight);
variableTime = [variable, '.time = ', num2str(Time), ';'];
evalin('base',variableTime);
variablegammaTOGA = [variable, '.gammaTOGA = ', num2str(Gamma), ';'];
evalin('base',variablegammaTOGA);
variablegammaLeksell = [variable, '.gammaLeksell = ', num2str(Gamma),
';'];
evalin('base',variablegammaLeksell);
variablehelmet = [variable, '.collimator_helmet = ', num2str(helmet),
';'];
evalin('base',variablehelmet);

guidata(hObject, handles);

function editTime_Callback(hObject, eventdata, handles)
% hObject      handle to editTime (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editTime as text
%         str2double(get(hObject,'String')) returns contents of editTime
%         as a double

% --- Executes during object creation, after setting all properties.
function editTime_CreateFcn(hObject, eventdata, handles)
% hObject      handle to editTime (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
%              called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

addnewshot(hObject, eventdata, handles);

function addnewshot(hObject, eventdata, handles)

```

```

% handles      structure with handles and user data (see GUIDATA)
%Sets the popupmenuPlan selection choices
%
%Function that saves shots position to handles.shots and planC

indexS = handles.planC{end};

%Get the currently selected shot
planindex = get(handles.popupmenuPlan, 'Value');

%Add the new shot
handles.planC{indexS.beamGeometry}(1,planindex).shots(1,end+1) =
handles.planC{indexS.beamGeometry}(1,planindex).shots(1,end);

%Update planC with the new shot
variable = ['planC{', num2str(indexS.beamGeometry), '}(1,',
num2str(planindex), ').shots(1,end+1) = planC{',
num2str(indexS.beamGeometry), '}(1,', num2str(planindex),
').shots(1,end);'];
evalin('base',variable);

for i = 1:length(handles.planC{handles.indexS.beamGeometry})
    handles.shots{i} =
handles.planC{handles.indexS.beamGeometry}(1,i).shots;
end
%Populate objects in the GUI
setGUIpopupmenuPlan(handles);
setGUIlistboxShots(handles);
seteditXYZ(handles);

guidata(hObject, handles);

% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject      handle to checkbox3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3

checkval = get(handles.checkbox3, 'Value');
if checkval == 1
    set(handles.textOriginal, 'String', 'Leksell');
else
    set(handles.textOriginal, 'String', 'Original');
end

```

StructMatGeometricCalc

```
function [StructMat] =  
StructMatGeometricCalc(StructMat,MatIndX,MatIndY,MatIndZ,shots,gamma,Pro  
files,RxIsodose);  
%  
%[StructMat] = StructMatGeometricCalc(StructMat,shot,RxIsodose);  
%  
%StructMat manipulation function  
%StructMat - Matrix of 1s and 0s representing the structure to be  
optimized  
%shots - [Helmet X Y Z] of shot to be applied to StructMat  
%RxIsodose - Isodose gaussian size parameter to represent coverage of  
shot  
%Profiles - Profiles on the shots  
%MatInd - Index of the structure matrix in X, Y, Z dimensions  
%  
% Joseph Bowling  
% University of Tennessee  
% 2011-05-07  
  
%Discretize shots - Round dose to nearest 0.05  
shots(:,1) = round(shots(:,1));  
shots(:,[2 3 4]) = round(shots(:,[2 3 4])*20)/20;  
  
%Setup the Dose Calculation  
numshots = size(shots,1);  
dcsize = size(StructMat);  
dose = zeros(size(StructMat),'single');  
  
%Calculate the dose  
for i = 1:numshots(1)  
  
    rshots = shots(i,2:4);  
    scx = find(abs(MatIndX-rshots(1)) <= 0.04);  
    scy = find(abs(MatIndY-rshots(2)) <= 0.04);  
    scz = find(abs(MatIndZ-rshots(3)) <= 0.04);  
  
    if length(scx) > 1  
        scx = scx(1);  
    end  
    if length(scy) > 1  
        scy = scy(1);  
    end  
    if length(scz) > 1  
        scz = scz(1);  
    end  
  
    if isempty(scx)  
        if rshots(1) > MatIndX(end)  
            extx = (rshots(1) - MatIndX(end))/0.05;  
            scx = length(MatIndX) + extx;  
            scx = single(scx);  
        elseif rshots(1) < MatIndX(1)  
            extx = (MatIndX(1) - rshots(1))/0.05;
```

```

        scx = 1 - extx;
        scx = single(scx);
    end
end
if isempty(scy)
    if rshots(2) > MatIndY(end)
        exty = (rshots(2) - MatIndY(end))/0.05;
        scy = length(MatIndY) + exty;
        scy = single(scy);
    elseif rshots(2) < MatIndY(1)
        exty = (MatIndY(1) - rshots(2))/0.05;
        scy = 1 - exty;
        scy = single(scy);
    end
end
if isempty(scz)
    if rshots(3) > MatIndZ(end)
        extz = (rshots(3) - MatIndZ(end))/0.05;
        scz = length(MatIndZ) + extz;
        scz = single(scz);
    elseif rshots(3) < MatIndZ(1)
        extz = (MatIndZ(1) - rshots(3))/0.05;
        scz = 1 - extz;
        scz = single(scz);
    end
end

%Select elements from dose calculation matrix
switch shots(i,1)
    case 4
        xedge = [scx-30 scx+30];
        yedge = [scy-30 scy+30];
        zedge = [scz-30 scz+30];

        pxedge = [1 61];
        pyedge = [1 61];
        pzedge = [1 61];
    case 8
        xedge = [scx-60 scx+60];
        yedge = [scy-60 scy+60];
        zedge = [scz-60 scz+60];

        pxedge = [1 121];
        pyedge = [1 121];
        pzedge = [1 121];
    case 14
        xedge = [scx-75 scx+75];
        yedge = [scy-75 scy+75];
        zedge = [scz-75 scz+75];

        pxedge = [1 151];
        pyedge = [1 151];
        pzedge = [1 151];
    case 18
        xedge = [scx-75 scx+75];
        yedge = [scy-75 scy+75];

```

```

        zedge = [scz-75 scz+75];

        pxedge = [1 151];
        pyedge = [1 151];
        pzedge = [1 151];
    end

    if xedge(1) <= 0
        diff = 1 - xedge(1);
        pxedge(1) = pxedge(1) + diff;
        xedge(1) = 1;
    end
    if yedge(1) <= 0
        diff = 1 - yedge(1);
        pyedge(1) = pyedge(1) + diff;
        yedge(1) = 1;
    end
    if zedge(1) <= 0
        diff = 1 - zedge(1);
        pzedge(1) = pzedge(1) + diff;
        zedge(1) = 1;
    end

    if xedge(2) > dcszsize(2)
        pxedge(2) = pxedge(2) - (xedge(2) - dcszsize(2));
        xedge(2) = dcszsize(2);
    end
    if yedge(2) > dcszsize(1)
        pyedge(2) = pyedge(2) - (yedge(2) - dcszsize(1));
        yedge(2) = dcszsize(1);
    end
    if zedge(2) > dcszsize(3)
        pzedge(2) = pzedge(2) - (zedge(2) - dcszsize(3));
        zedge(2) = dcszsize(3);
    end

    evalstring =
    ['dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) =
dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) +
shots(i,5) *
Profiles.H',num2str(shots(i,1)),'.G',num2str(gamma),'(pyedge(1):pyedge(
2),pxedge(1):pxedge(2),pzedge(1):pzedge(2));'];
    %evalstring =
    ['dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) =
dose(yedge(1):yedge(2),xedge(1):xedge(2),zedge(1):zedge(2)) +
Profiles.H',num2str(shots(i,1)),'.G',num2str(gamma),'(pyedge(1):pyedge(
2),pxedge(1):pxedge(2),pzedge(1):pzedge(2));'];
    eval(evalstring);

end

%Normalize the dose matrix to unity
dose = dose ./ max(max(max(dose)));

%Find all elements of the dose matrix that are greater than RxIsodose
a = find(dose > RxIsodose);

```

```
%Zero all elements of StructMat that correspond to elements of dose  
that  
%are greater than RxIsodose  
StructMat(a) = 0;
```

TOGAGAOptFunc

```
function [Cost] =
TOGAGAOptFunc(para,planC,doseindex,Profiles,gamma,w,structNum,RxIsodose
,TV,CalcMethod);
%
%[Cost] =
TOGAGAOptFunc(para,planC,doseindex,Profiles,gamma,w,structNum,RxIsodose
,TV,CalcMethod);
%
%Cost function for optimization of Gamma Knife treatment planning with
%Genetic Algorithms.
%
%para - parametered supplied by the GA
%doseindex - dose index for optimization
%gamma - gamma angle for optimization
%structnum - structure to be optimized. The first row of the variable
is
%the only required row and represents the structure to be optimized.
If
%the variable has two rows the second row is the structure index for
the
%ring around the optimization volume to be optimized. Rows three and
%greater represent critical structures and the second column of this
row
%represents the maximum dose to the structure.
%RxIsodose - Rx isodose for optimization
%TV - Volume of the structure to be optimized
%CalcMethod - Dose calc method
%
%Written by Joseph Bowling University of Tennessee 3/4/2011

%Ensure all weights are positive in the para variable
para(find(para<=0.05)) = 0;

%Calculate the dose distribution with the shot arrangement provided by
the
%genetic algorithm
[planC] = dosecalcTOGA(planC,doseindex,Profiles,CalcMethod,para,gamma);

%Determine the quality of the plan by the conformity indices
if size(structNum,1) >= 2;
    if structNum(2,1) ~= 0

[Iso_Cov,Cov_RTog,CI_RTog,CI_Paddick,CI_Gradient,CI_Karolinska,RingInde
x] = ConformIndex(planC,structNum,doseindex,RxIsodose,TV);
    else
        [Iso_Cov,Cov_RTog,CI_RTog,CI_Paddick,CI_Gradient,CI_Karolinska]
= ConformIndex(planC,structNum,doseindex,RxIsodose,TV);
    end
else
    [Iso_Cov,Cov_RTog,CI_RTog,CI_Paddick,CI_Gradient,CI_Karolinska] =
ConformIndex(planC,structNum,doseindex,RxIsodose,TV);
end
%Manipulate the conformity indices to make them minimize function with
%values between 0 and 1
```



```

CI(1) = (-1*Iso_Cov)+1; %Iso_Cov has values 0 - 1 with 1 being perfect
CI(2) = (-1*Cov_RTOG)+1;%Cov_RTOG is minimum dose to target, want it to
be
                                %greater than 0.9 with 1.0 being perfect
CI(3) = CI_RTOG - 1.2; %CI_RTOG has values 1 and greater with 1 being
perfect
                                %and less than two being acceptable. I chose
- 1.2
                                %based on experience.
CI(4) = (-1*CI_Paddick)+1; %Has values between 0 and 1 with 1 being
perfect
CI(5) = abs(CI_Gradient - 2.5);%Two is perfect
CI(6) = CI_Karolinska - 3;%Should be less than 4
%Sum of square error optimization
if size(structNum,1) >= 2;
    if structNum(2,1) ~= 0
        CI(7) = RingIndex;
    else
        CI(7) = 0;
    end
else
    CI(7) = 0;
end

%Critical Structure Optimization
if size(structNum,1) >= 3;
    [dosesV, volsV, isError] = getDVH(structNum(3,1), doseindex,
planC);
    f = find(dosesV>structNum(3,2));
    if length(f) == 0
        CI(8) =0;
    else
        CI(8) = sum((dosesV(f)-structNum(3,2)).^2)/length(dosesV(f));
    end
else
    CI(8) = 0;
end

%Sum the weighted conformity indices to determine the final cost
Cost = sum (w .* CI) / sum(w);

```

TOGAGAOptFuncHybrid

```
function [Cost] =
TOGAGAOptFuncHybrid(para,planC,doseindex,Profiles,gamma,w,structNum,RxI
sodose,TV,CalcMethod,Collimator);
%
%[Cost] =
TOGAGAOptFuncHybrid(para,planC,doseindex,Profiles,gamma,w,structNum,RxI
sodose,TV,CalcMethod,Collimator);
%
%Cost function for optimization of Gamma Knife treatment planning with
%hybrid function post GA optimization.
%
%para - parametered supplied by the GA
%doseindex - dose index for optimization
%gamma - gamma angle for optimization
%structnum - structure to be optimized. The first row of the variable
is
%the only required row and represents the structure to be optimized.
If
%the variable has two rows the second row is the structure index for
the
%ring around the optimization volume to be optimized. Rows three and
%greater represent critical structures and the second column of this
row
%represents the maximum dose to the structure.
%RxIsodose - Rx isodose for optimization
%TV - Volume of the structure to be optimized
%CalcMethod - Dose calc method
%Collimator - The binary representation of the collimator from the
%optimization
%
%Written by Joseph Bowling University of Tennessee 11/11/2011

%Revise the para variable to fit the calculation and include the
collimator
numshots = size(Collimator,2)/2;
holder = [];
countc = 1;
counts = 1;
for i = 1:numshots

    holder = [holder Collimator(1,[countc countc+1])
para(1,[counts:counts+3])];
    countc = countc + 2;
    counts = counts + 4;
end
para = holder;

%Ensure all weights are positive in the para variable
para(find(para<=0.05)) = 0;

%Calculate the dose distribution with the shot arrangement provided by
the
%genetic algorithm
```

```

[planC] = dosecalcTOGA(planC,doseindex,Profiles,CalcMethod,para,gamma);

%Determine the quality of the plan by the conformity indices
if size(structNum,1) >= 2;

[Iso_Cov,Cov_RTOG,CI_RTOG,CI_Paddick,CI_Gradient,CI_Karolinska, RingIndex] = ConformIndex(planC, structNum, doseindex, RxIsodose, TV);
else
    [Iso_Cov,Cov_RTOG,CI_RTOG,CI_Paddick,CI_Gradient,CI_Karolinska] = ConformIndex(planC, structNum, doseindex, RxIsodose, TV);
end
%Manipulate the conformity indices to make them minimize function with values between 0 and 1
CI(1) = (-1*Iso_Cov)+1; %Iso_Cov has values 0 - 1 with 1 being perfect
CI(2) = (-1*Cov_RTOG)+1;%Cov_RTOG is minimum dose to target, want it to be
%greater than 0.9 with 1.0 being perfect
CI(3) = CI_RTOG - 1.2; %CI_RTOG has values 1 and greater with 1 being perfect
%and less than two being acceptable. I chose
- 1.2
%based on experience.
CI(4) = (-1*CI_Paddick)+1; %Has values between 0 and 1 with 1 being perfect
CI(5) = CI_Gradient - 2;%Two is perfect
CI(6) = CI_Karolinska - 3;%Should be less than 4
%Sum of square error optimization
if size(structNum,1) >= 2;
    CI(7) = RingIndex;
else
    CI(7) = 0;
end

%Critical Structure Optimization
if size(structNum,1) >= 3;
    [dosesV, volsV, isError] = getDVH(structNum(3,1), doseindex, planC);
    f = find(dosesV>structNum(3,2));
    if length(f) == 0
        CI(8) =0;
    else
        CI(8) = sum((dosesV(f)-structNum(3,2)).^2)/length(dosesV(f));
    end
else
    CI(8) = 0;
end

%Sum the weighted conformity indices to determine the final cost
Cost = sum (w .* CI) / sum(w);

```

TOGAGUI

```
function varargout = TOGAGUI(varargin)
% TOGAGUI M-file for TOGAGUI.fig
%     TOGAGUI, by itself, creates a new TOGAGUI or raises the existing
%     singleton*.
%
%     H = TOGAGUI returns the handle toga_utilities_dosecalc a new
%     TOGAGUI or the handle toga_utilities_dosecalc
%     the existing singleton*.
%
%     TOGAGUI('CALLBACK',hObject,eventData,handles,...) calls the
%     local
%     function named CALLBACK in TOGAGUI.M with the given input
%     arguments.
%
%     TOGAGUI('Property','Value',...) creates a new TOGAGUI or raises
%     the
%     existing singleton*. Starting from the left, property value
%     pairs are
%     applied toga_utilities_dosecalc the GUI before
%     TOGAGUI_OpeningFunction gets called. An
%     unrecognized property name or invalid value makes property
%     application
%     stop. All inputs are passed toga_utilities_dosecalc
%     TOGAGUI_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
%     one
%     instance toga_utilities_dosecalc run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text toga_utilities_dosecalc modify the response
toga_utilities_dosecalc help TOGAGUI

% Last Modified by GUIDE v2.5 06-Dec-2011 07:18:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @TOGAGUI_OpeningFcn, ...
                  'gui_OutputFcn',  @TOGAGUI_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

% End initialization code - DO NOT EDIT

% --- Executes just before TOGAGUI is made visible.
function TOGAGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle toga_utilities_dosecalc figure
% eventdata  reserved - toga_utilities_dosecalc be defined in a future
version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin    command line arguments toga_utilities_dosecalc TOGAGUI
(see VARARGIN)

% Choose default command line output for TOGAGUI
handles.output = hObject;

warning off;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TOGAGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned toga_utilities_dosecalc
the command line.
function varargout = TOGAGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle toga_utilities_dosecalc figure
% eventdata  reserved - toga_utilities_dosecalc be defined in a future
version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% -----
function TOGA_Utilityies_File_Callback(hObject, eventdata, handles)
% hObject    handle toga_utilities_dosecalc TOGA_Utilityies_File (see
GCBO)
% eventdata  reserved - toga_utilities_dosecalc be defined in a future
version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% -----
function TOGA_Utilityies_Database_Callback(hObject, eventdata, handles)
% hObject    handle toga_utilities_dosecalc TOGA_Utilityies_Database
(see GCBO)
% eventdata  reserved - toga_utilities_dosecalc be defined in a future
version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% -----
function TOGA_Utilityies_DoseCalc_Callback(hObject, eventdata, handles)
% hObject      handle toga_utilityies_dosecalc TOGA_Utilityies_DoseCalc
% (see GCBO)
% eventdata    reserved - toga_utilityies_dosecalc be defined in a future
version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% -----
function TOGA_Utilityies_TOGA_Utilityies_Callback(hObject, eventdata,
handles)
% hObject      handle to TOGA_Utilityies_TOGA_Utilityies (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% -----
function TOGA_Utilityies_Callback(hObject, eventdata, handles)
% hObject      handle to TOGA_Utilityies (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% -----
function File_OpenPatient_Callback(hObject, eventdata, handles)
% hObject      handle to File_OpenPatient (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

[FileName,PathName] = uigetfile('*.mat','Select the planC mat file to
load');
if FileName == 0
    return
end
Path = [PathName FileName];
Command = ['load(''', Path, ''')'];
evalin('base',Command);
evalin('base','global planC');

% -----
function File_FileInfo_Callback(hObject, eventdata, handles)
% hObject      handle to File_FileInfo (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found.  Please use File->Open Patient
to load file.\n\n')
else
    openvar('planC')

```

end

```
% -----  
function File_DICOMRT_Callback(hObject, eventdata, handles)  
% hObject    handle to File_DICOMRT (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
  
% -----  
function File_GammaPlan_Callback(hObject, eventdata, handles)  
% hObject    handle to File_GammaPlan (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
  
str = computer;  
if strcmp(str, 'PCWIN')  
    [planDir] = uigetdir('C:\TOGA\Patient Data\','Select the directory  
where the GammaPlan files to be imported reside');  
elseif strcmp(str, 'MACI64')  
    [planDir] = uigetdir('/TOGA/Patient Data/','Select the directory  
where the GammaPlan files to be imported reside');  
end  
  
if planDir == 0  
    return  
end  
[planC, leksellInfo] = importLeksellPlan(planDir);  
planC{indexS.gammaknife} = leksellInfo;  
planC = setUniformizedData(planC);  
  
%Save planC in base work  
assignin('base', 'planC', planC);  
  
%Save the file  
reply = input('\n\nWould you like to save the imported plan? ', 's');  
matstring = strmatch(reply, strvcat('Yes', 'Y', 'yes', 'YES'));  
if ~isempty(matstring)  
    if strcmp(str, 'PCWIN')  
        [saveDir] = uigetdir('C:\TOGA\Patient Data\','Select the  
directory to save the patient file');  
        fprintf(1, 'Saving planC variable.\n\n');  
        strplace = strfind(planDir, '\');  
        filename = planDir(strplace(end):length(planDir));  
        filename = [saveDir filename, '.mat'];  
        savecommand = ['save(''', filename, ''', 'planC')'];  
        evalin('base', savecommand);  
    elseif strcmp(str, 'MACI64')  
        [saveDir] = uigetdir('/TOGA/Patient Data/','Select the  
directory to save the patient file');  
        fprintf(1, 'Saving planC variable.\n\n');  
        strplace = strfind(planDir, '/');  
        filename = planDir(strplace(end):length(planDir));  
        filename = [saveDir filename, '.mat'];
```

```

        savecommand = ['save('', filename, '', 'planC')'];
        evalin('base', savecommand);
    end

end

% -----
function DoseCalc_NewDoseGrid_Callback(hObject, eventdata, handles)
% hObject      handle to DoseCalc_NewDoseGrid (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

evalin('base', 'global planC');
handles.planC = evalin('base', 'planC');
if isempty(handles.planC)
    fprintf(1, 'PlanC variable not found. Please use File->Open Patient
to load file before running the shots GUI.\n\n')
else
    PlanGUI
end

% -----
function DoseCalc_Shots_Callback(hObject, eventdata, handles)
% hObject      handle to DoseCalc_Shots (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

evalin('base', 'global planC');
handles.planC = evalin('base', 'planC');
if isempty(handles.planC)
    fprintf(1, 'PlanC variable not found. Please use File->Open Patient
to load file before running the shots GUI.\n\n')
else
    ShotsGUI
end

% -----
function DoseCalc_DoseCalc_Callback(hObject, eventdata, handles)
% hObject      handle to DoseCalc_DoseCalc (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

evalin('base', 'global planC');
handles.planC = evalin('base', 'planC');
if isempty(handles.planC)
    fprintf(1, 'PlanC variable not found. Please use File->Open Patient
to load file before running the shots GUI.\n\n')
else
    DoseCalcGUI
end

% -----
function TOGA_Optimization_Callback(hObject, eventdata, handles)
% hObject      handle to TOGA_Optimization (see GCBO)

```



```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

OptGATOGAGui

```
% -----
function Utilities_LeksellFiducials_Callback(hObject, eventdata,
handles)
% hObject handle to Utilities_LeksellFiducials (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% -----
function Utilities_Isocenter_Callback(hObject, eventdata, handles)
% hObject handle to Utilities_Isocenter (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% -----
function Utilities_Profiles_Callback(hObject, eventdata, handles)
% hObject handle to Utilities_Profiles (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% -----
function TOGA_File_Callback(hObject, eventdata, handles)
% hObject handle to TOGA_File (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% -----
function TOGA_Database_Callback(hObject, eventdata, handles)
% hObject handle to TOGA_Database (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% -----
function TOGA_DoseCalc_Callback(hObject, eventdata, handles)
% hObject handle to TOGA_DoseCalc (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)z
```

```
% -----
function TOGA_TOGA_Callback(hObject, eventdata, handles)
% hObject handle to TOGA_TOGA (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```

% -----
function File_Convert2Cerr_Callback(hObject, eventdata, handles)
% hObject    handle to File_Convert2Cerr (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Convert2Cerr

% -----
function File_SavePatient_Callback(hObject, eventdata, handles)
% hObject    handle to File_SavePatient (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%See if planC exists
evalin('base','global planC');
handles.planC = evalin('base','planC');
if isempty(handles.planC)
    fprintf(1,'PlanC variable not found.  Please use File->Open Patient
to load file.\n\n\n\n')
    return
else
    str = computer;
    if strcmp(str, 'PCWIN')
        [FileName,PathName] = uiputfile('*.mat','Enter the save path
name','C:\TOGA\Patient Data\');
    elseif strcmp(str, 'MACI64')
        [FileName,PathName] = uiputfile('*.mat','Enter the save path
name','/TOGA/Patient Data/');
    end
    if FileName == 0
        return
    end
    SavePath = [PathName FileName];

    %Save the file
    savecommand = ['save(''', SavePath, '', 'planC')'];
    evalin('base',savecommand);
end

% -----
function DoseCalc_Report_Callback(hObject, eventdata, handles)
% hObject    handle to DoseCalc_Report (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

ReportGUI

% -----
function ConformityIndex_Callback(hObject, eventdata, handles)
% hObject    handle to ConformityIndex (see GCBO)

```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
```

ConformityGUI

```
% -----
function importX02PlanC_Callback(hObject, eventdata, handles)
% hObject      handle to importX02PlanC (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
```

ImportX0GUI

```
% -----
function TOGA_GAInitPop_Callback(hObject, eventdata, handles)
% hObject      handle to TOGA_GAInitPop (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
```

GAInitialPopulationGUI

TransformGAOptFunc

```
function [Cost] =
TransformGAOptFunc(para, StructMat, MatIndX, MatIndY, MatIndZ, gamma, Profile
s, RxIsodose, TransformShots, VolInit);
%
%[Cost] =
TransformGAOptFunc(para, StructMat, MatIndX, MatIndY, MatIndZ, gamma, Profile
s, RxIsodose, TransformShots, VolInit);
%
%Cost function for the geometric optimization of the transform of the
prior selected plans with
%Genetic Algorithms.
%
%Written by Joseph Bowling University of Tennessee 6/27/2011

%Calculate the geometric coverage with the shot arrangement provided by
the
%genetic algorithm
%Para is a 6 component vector representing the translation and rotation
of
%the original plan.  Components 1-3 are the translation and 3:6 are the
3D
%rotation

Translation = para(1:3);
alpha = para(4);
beta = para(5);
gam = para(6);
Rx = [1 0 0;0 cos(alpha) sin(alpha);0 -sin(alpha) cos(alpha)];
Ry = [cos(beta) 0 -sin(beta);0 1 0;sin(beta) 0 cos(beta)];
Rz = [cos(gam) sin(gam) 0;-sin(gam) cos(gam) 0;0 0 1];

%Center the shots around zero before rotating
Shift2ZeroCen = mean(TransformShots(:,2:4));
TransformShots(:,2:4) = TransformShots(:,2:4) -
repmat(Shift2ZeroCen,size(TransformShots,1),1);

%Roate in X
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) * Rx];
end
%Rotate in Y
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) * Ry];
end
%Rotate in Z
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) * Rz];
end

%Shift back to prior location
TransformShots(:,2:4) = TransformShots(:,2:4) +
repmat(Shift2ZeroCen,size(TransformShots,1),1);
```

```

%Translation
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) + Translation];
end

[StructMat] =
StructMatGeometricCalc(StructMat,MatIndX,MatIndY,MatIndZ,TransformShots,
gamma,Profiles,RxIsodose);

%Calculated the portion of the target not covered
Cost = sum(sum(sum(StructMat)))/VolInit;

```

TransformPlan

```
function [TransformShots] = TransformPlan(planC, StructMat, structNum,
doseindex, gamma, method, MatIndX, MatIndY, MatIndZ, shots,
CenterContour);
%
%[TransformShots] = TransformPlan(planC, StructMat, structNum,
doseindex, gamma, method, MatIndX, MatIndY, MatIndZ, shots,
CenterContour);
%
%Transform the selected similar plans to the structure to be optimized
%TransformShots - The transformed set of shots
%shots - [Helmet X Y Z Weight] of shot to be transformed
%CenterContour - the geometric center of the structure the shots are to
be
%transformed to
%MatInd - Index of the structure matrix in X, Y, Z dimensions
%
% Joseph Bowling
% University of Tennessee
% 2011-06-24

%Shift the selected prior plans to align with the current optimization
%volume
%Calculate shift to center
ShotsCenter = [mean(shots(:,2)) mean(shots(:,3)) mean(shots(:,4))];
Transform = CenterContour - ShotsCenter;
TransformShots = shots;
TransformShots(:, [2:4]) = TransformShots(:, 2:4) +
repmat (Transform, size (TransformShots, 1), 1);

%Calculate initial volume of structure to be fitted
VolInit = sum(sum(sum(StructMat)));

%Load Profiles
[Profiles] = loadprofilesTOGA(planC, doseindex, gamma);

%Define the optimization function
OptFunc =
@ (para) TransformGAOptFunc (para, StructMat, MatIndX, MatIndY, MatIndZ, gamma,
Profiles, RxIsodose, TransformShots, VolInit);

%Setup the GA optimization
nvars = 6;
PopInitRange_Data = [-2 -2 -2 0 0 0; 2 2 2 2*pi 2*pi 2*pi];
PopulationSize_Data = 12;
Generations_Data = 10;
lb = [-2 -2 -1 0 0 0];
ub = [2 2 1 2*pi 2*pi 2*pi];

% Start with the default options
options = gaoptimset;
% Modify options setting
options = gaoptimset(options, 'PopInitRange', PopInitRange_Data);
options = gaoptimset(options, 'PopulationSize', PopulationSize_Data);
```

```

options = gaoptimset(options,'Generations', Generations_Data);
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', { @gaplotbestf @gaplotscores
});
%options = gaoptimset(options,'UseParallel', 'always');

%Run the actual Genetic Algorithm Optimization
[x,fval,exitflag,output,population,score] = ...
ga(@ (para)TransformGAOptFunc(para,StructMat,MatIndX,MatIndY,MatIndZ,gam
ma,Profiles,0.5,TransformShots,VolInit),nvars,[],[],[],[],lb,ub,[],opti
ons);

%Transform the shots with the optimal transformation
Translation = x(1:3);
alpha = x(4);
beta = x(5);
gam = x(6);
Rx = [1 0 0;0 cos(alpha) sin(alpha);0 -sin(alpha) cos(alpha)];
Ry = [cos(beta) 0 -sin(beta);0 1 0;sin(beta) 0 cos(beta)];
Rz = [cos(gam) sin(gam) 0;-sin(gam) cos(gam) 0;0 0 1];

%Center the shots around zero before rotating
Shift2ZeroCen = mean(TransformShots(:,2:4));
TransformShots(:,2:4) = TransformShots(:,2:4) -
repmat(Shift2ZeroCen,size(TransformShots,1),1);

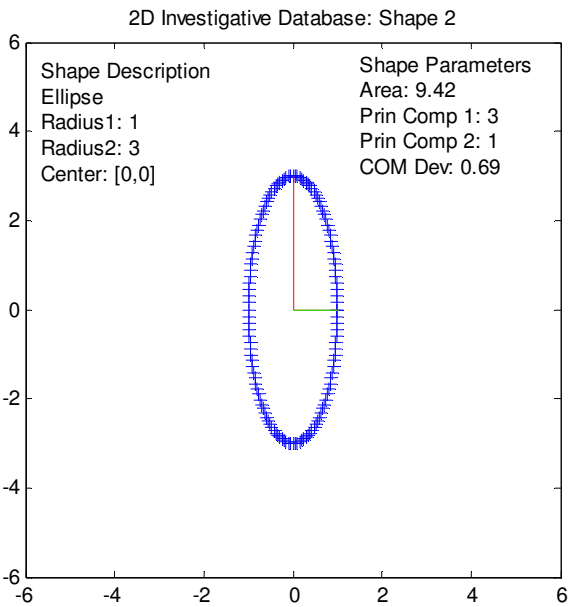
%Rotate in X
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) * Rx];
end
%Rotate in Y
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) * Ry];
end
%Rotate in Z
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) * Rz];
end

%Shift back to prior location
TransformShots(:,2:4) = TransformShots(:,2:4) +
repmat(Shift2ZeroCen,size(TransformShots,1),1);

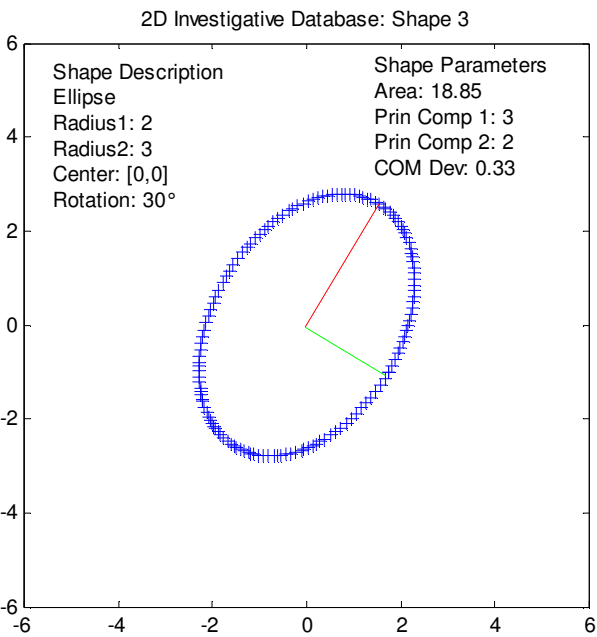
%Translation
for i = 1:size(TransformShots,1)
    TransformShots(i,2:4) = [TransformShots(i,2:4) + Translation];
end

```

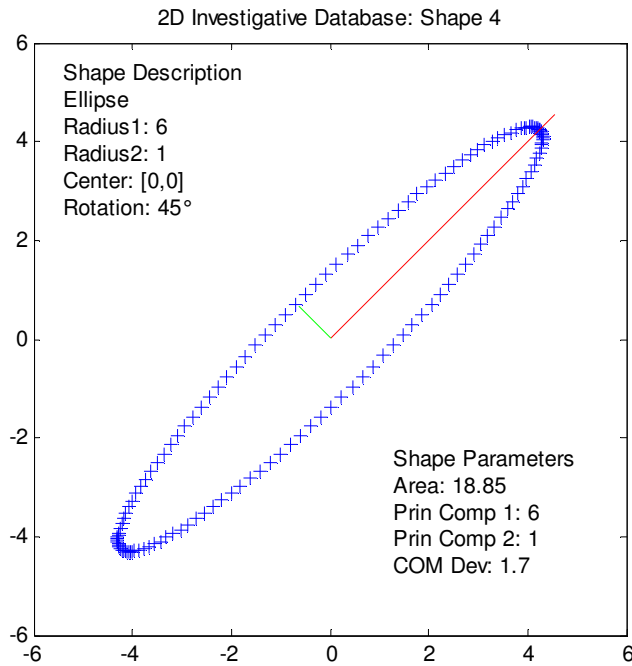
Appendix B: Figures and Tables



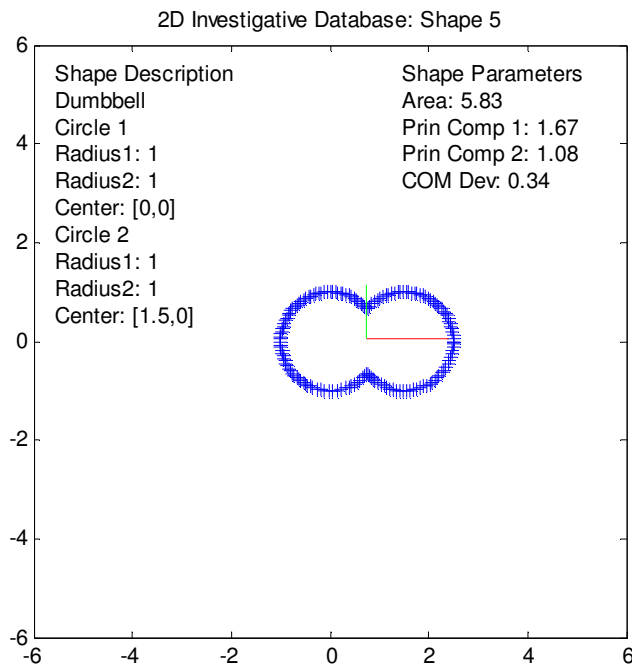
Appendix Figure 54 Investigation Database Shape 2



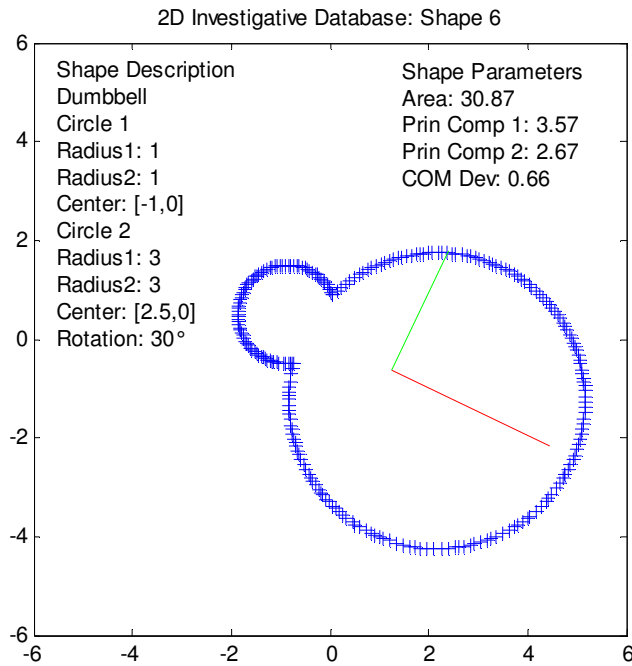
Appendix Figure 55 Investigation Database Shape 3



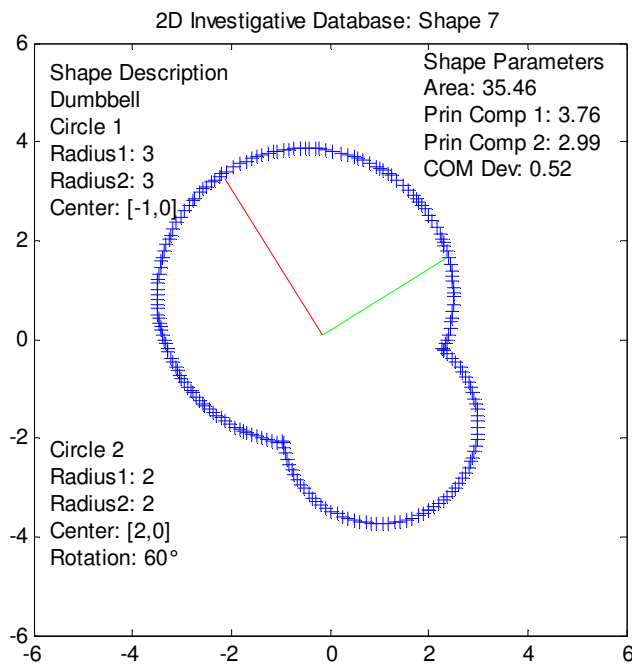
Appendix Figure 56 Investigation Database Shape 4



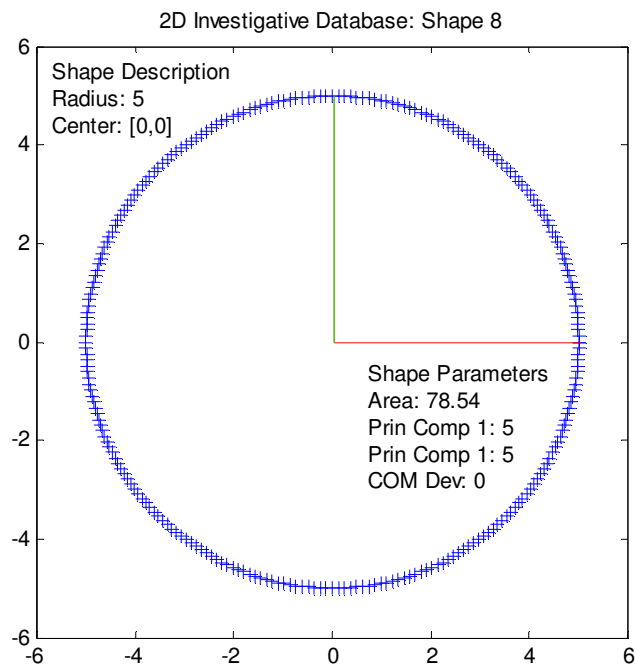
Appendix Figure 57 Investigation Database Shape 5



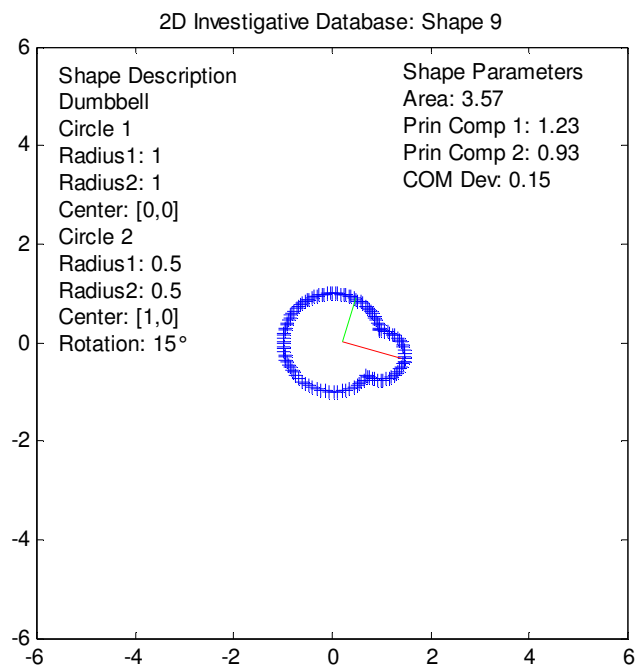
Appendix Figure 58 Investigation Database Shape 6



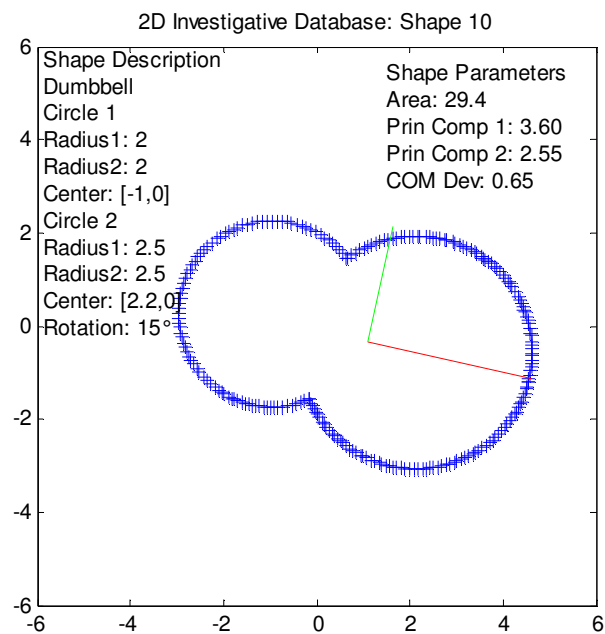
Appendix Figure 59 Investigation Database Shape 7



Appendix Figure 60 Investigation Database Shape 8



Appendix Figure 61 Investigation Database Shape 9



Appendix Figure 62 Investigation Database Shape 10

Appendix Table 11 Historical Database Treatment Volume Table

Structure Index	Tumor Volume	Variance of COM to edge dist.	Variance of COM to edge dist.	Principal Component 1	Principal Component 2	Principal Component 3
1	2.8446	0.0227	0.0227	1.831	1.3888	1.2121
2	3.3807	0.1124	0.1124	2.5123	1.5582	1.0513
3	3.9216	0.0483	0.0483	2.1034	1.5323	1.3272
4	4.5231	0.0116	0.0116	1.9233	1.7875	1.3479
5	0.9907	0.0181	0.0181	1.2336	1.0886	0.8104
6	2.106	0.0234	0.0234	1.714	1.153	1.1331
7	6.0858	0.0556	0.0556	2.1333	2.0376	1.609
8	0.3515	0.0139	0.0139	0.9781	0.8479	0.5407
9	5.7035	0.0176	0.0176	2.0677	1.8125	1.606
10	9.6217	0.0844	0.0844	2.6737	2.3949	1.7282
11	5.6805	0.0829	0.0829	2.6355	2.1154	1.176
12	1.6153	0.0127	0.0127	1.4675	1.2923	0.9303
13	5.1699	0.0235	0.0235	2.0892	1.8323	1.4823
14	1.6218	0.0437	0.0437	1.9063	1.4709	0.5686
15	15.5012	0.0331	0.0331	2.9652	2.5369	2.1932
16	1.5155	0.0315	0.0315	1.6172	1.4717	0.7483
17	3.6625	0.047	0.047	1.8142	1.7583	1.3754
18	0.9705	0.0191	0.0191	1.3188	1.1293	0.8849
19	3.7185	0.0502	0.0502	2.299	2.0034	0.8918
20	0.6981	0.0099	0.0099	1.1802	0.991	0.7375
21	18.607	0.0396	0.0396	3.0486	2.6917	2.3837
22	3.8373	0.0505	0.0505	2.0516	1.6138	1.4162
23	0.3573	0.0064	0.0064	0.9367	0.8475	0.5157
24	1.0687	0.0382	0.0382	1.6514	1.0493	0.7292
25	1.4289	0.0382	0.0382	1.5286	1.3509	0.8093
26	0.5354	0.0289	0.0289	1.3431	0.795	0.5408
27	0.2426	0.0125	0.0125	0.9117	0.6113	0.5407
28	4.3325	0.0364	0.0364	2.1837	1.5982	1.3202
29	3.9027	0.0164	0.0164	1.9784	1.5792	1.3383
30	6.3571	0.0743	0.0743	2.6455	1.8413	1.575
31	8.155	0.1698	0.1698	3.1928	2.077	1.4705
32	2.8554	0.0391	0.0391	1.915	1.3778	1.1589
33	0.0716	0.0065	0.0065	0.7912	0.669	0.1353
34	7.2743	0.0088	0.0088	2.2232	2.0157	1.6482
35	8.7803	0.0571	0.0571	2.5549	2.07	1.8756
36	13.7266	0.4543	0.4543	4.0422	2.6177	1.8366
37	1.7436	0.0344	0.0344	1.7266	1.1683	0.9827
38	0.9139	0.0233	0.0233	1.2992	1.0426	0.8089
39	3.7055	0.2013	0.2013	2.9702	1.7425	0.8865

Structure Index	Tumor Volume	Variance of COM to edge dist.	Variance of COM to edge dist.	Principal Component 1	Principal Component 2	Principal Component 3
40	3.0922	0.1745	0.1745	2.4751	1.7054	1.5095
41	2.1377	0.0279	0.0279	1.7067	1.5032	0.9988
42	16.4088	0.0409	0.0409	2.9183	2.5577	2.4805
43	2.6163	0.073	0.073	2.3264	1.376	1.0025
44	1.5413	0.0296	0.0296	1.8281	1.0906	0.9104
45	16.6069	0.0343	0.0343	3.0761	2.5524	2.2384
46	5.3118	0.0773	0.0773	2.6051	2.0448	1.3383
47	3.0009	0.1653	0.1653	2.9518	1.1804	1.0534
48	5.57	0.0085	0.0085	2.087	1.9282	1.4198
49	1.4797	0.0085	0.0085	1.3272	1.2617	1.0134
50	1.127	0.0493	0.0493	1.4414	1.3486	1.0912
51	4.5297	0.0039	0.0039	1.8517	1.7947	1.3938
52	2.579	0.0914	0.0914	2.3962	1.7025	0.8677
53	10.4192	0.0678	0.0678	2.9569	2.0706	1.8181
54	11.6697	0.0557	0.0557	3.0912	2.3652	2.1881
55	27.7877	0.2747	0.2747	4.2825	3.2026	2.5264
56	5.819	0.0207	0.0207	2.0955	1.8167	1.6273
57	4.8595	0.0412	0.0412	2.2293	2.0553	1.1721
58	0.6643	0.0062	0.0062	1.0145	0.9765	0.7354
59	2.1885	0.024	0.024	1.7427	1.4597	0.9362
60	21.009	0.2059	0.2059	4.1422	2.791	2.1887
61	7.2235	0.0169	0.0169	2.3435	2.035	1.5906
62	0.3573	0.0064	0.0064	0.9367	0.8475	0.5157
63	0.439	0.0218	0.0218	1.2172	0.8317	0.6399
64	19.3678	0.1756	0.1756	3.808	2.7642	2.2726
65	0.3305	0.0177	0.0177	1.1656	0.6619	0.5002
66	0.4827	0.0106	0.0106	1.0655	0.8005	0.6584
67	8.5583	0.0414	0.0414	2.4606	2.0494	1.8679
68	0.8932	0.0115	0.0115	1.1602	1.03	0.8564
69	9.4469	0.0501	0.0501	2.8844	2.089	1.5822
70	4.5878	0.0795	0.0795	2.2309	1.6721	1.2735
71	3.1216	0.0297	0.0297	2.1814	1.6335	1.1358
72	5.2782	0.1191	0.1191	2.8347	1.9986	1.0409
73	1.7684	0.0442	0.0442	1.9135	1.3157	0.9935
74	4.1303	0.0346	0.0346	1.9086	1.5367	1.4435
75	5.0648	0.0864	0.0864	2.4848	1.8727	1.3948
76	0.3844	0.0313	0.0313	1.2475	0.6556	0.631
77	8.4827	0.4123	0.4123	3.9905	3.3181	1.8273
78	2.3394	0.032	0.032	1.9471	1.464	0.9142
79	7.552	0.1065	0.1065	2.9269	2.1986	1.6747
80	5.2438	0.0454	0.0454	2.3171	1.7188	1.4838
81	6.4838	0.0364	0.0364	2.3706	1.7523	1.6256

Structure Index	Tumor Volume	Variance of COM to edge dist.	Variance of COM to edge dist.	Principal Component 1	Principal Component 2	Principal Component 3
82	2.9495	0.0473	0.0473	1.901	1.3017	1.2735
83	2.9495	0.0473	0.0473	1.901	1.3017	1.2735
84	1.0677	0.039	0.039	1.3797	1.1611	0.9804
85	2.4035	0.0375	0.0375	1.8256	1.6556	0.9063
86	4.2732	0.5382	0.5382	5.0925	1.3646	0.9486
87	1.194	0.0062	0.0062	1.3205	1.1456	0.8341
88	6.0778	0.5222	0.5222	4.3183	2.1041	1.0916
89	2.4596	0.0665	0.0665	2.1796	1.7358	1.0832
90	1.7003	0.1055	0.1055	1.8496	1.4362	1.023
91	25.5321	1.2008	1.2008	6.2507	2.9414	2.2665
92	5.2878	0.0164	0.0164	2.0815	1.7501	1.5055
93	2.0525	0.0084	0.0084	1.5709	1.3922	0.9747
94	0.3043	0.0176	0.0176	1.1237	0.7412	0.4024
95	0.4535	0.0071	0.0071	1.075	0.9292	0.5138
96	6.123	0.0642	0.0642	2.3384	2.0815	1.4785
97	2.1947	0.0124	0.0124	1.6403	1.2933	1.1406
98	8.724	0.0748	0.0748	2.945	2.198	1.3881
99	4.5324	0.0316	0.0316	1.8661	1.7894	1.5141
100	4.4902	0.0404	0.0404	2.0321	1.6205	1.4145
101	13.7266	0.4543	0.4543	4.0422	2.6177	1.8366
102	4.5878	0.0795	0.0795	2.2309	1.6721	1.2735
103	8.8393	0.0119	0.0119	2.3531	2.0571	1.9385
104	2.8422	0.0239	0.0239	1.912	1.4442	1.0945
105	1.5519	0.0154	0.0154	1.6064	1.151	1.0155
106	1.0349	0.0238	0.0238	1.4709	0.9357	0.8461
107	1.9677	0.0929	0.0929	2.12	1.4604	0.7561
108	5.212	0.052	0.052	2.3908	1.8189	1.3897
109	0.2149	0.0096	0.0096	0.923	0.5836	0.4331
110	7.0325	0.0238	0.0238	2.2171	1.9524	1.77
111	0.4038	0.0058	0.0058	1.0502	0.7345	0.5514
112	0.2721	0.012	0.012	0.9493	0.5804	0.5338
113	8.7766	0.0938	0.0938	2.8581	1.984	1.6838
114	4.9362	0.3477	0.3477	3.9616	1.6471	1.3291
115	0.632	0.0178	0.0178	1.2886	0.8778	0.6126
116	3.3	0.0913	0.0913	2.2141	1.735	1.0701
117	4.6035	0.1473	0.1473	2.693	1.6782	1.3655
118	1.7063	0.0896	0.0896	2.2398	1.4107	0.8227
119	2.7226	0.0447	0.0447	1.7988	1.4554	1.2981
120	10.1086	0.0603	0.0603	2.8557	2.1089	1.8091
121	0.3844	0.0225	0.0225	1.2972	0.6659	0.502
122	12.1121	0.449	0.449	4.297	2.6701	1.446
123	2.281	0.0086	0.0086	1.5472	1.4142	1.0897

Structure Index	Tumor Volume	Variance of COM to edge dist.	Variance of COM to edge dist.	Principal Component 1	Principal Component 2	Principal Component 3
124	7.5003	0.053	0.053	2.346	2.1158	1.6318
125	20.8543	0.171	0.171	4.031	2.6008	2.1981
126	3.5367	0.0562	0.0562	2.2038	1.7189	1.1685
127	0.4015	0.022	0.022	1.1896	1.0813	0.4507
128	0.8487	0.0984	0.0984	2.223	0.9454	0.632
129	6.9215	0.3575	0.3575	3.842	2.1047	1.3156
130	15.8197	0.2702	0.2702	3.6084	3.0627	1.9016
131	7.9856	0.0196	0.0196	2.1514	2.0828	1.8329
132	2.6017	0.0124	0.0124	1.5675	1.417	1.2358
133	5.8274	0.0548	0.0548	2.2042	2.0199	1.5555
134	7.5472	0.0791	0.0791	2.9706	2.5287	1.0885
135	21.9341	0.061	0.061	3.2787	2.9778	2.5366
136	1.9568	0.0385	0.0385	1.5236	1.3043	0.9664
137	5.6413	0.0428	0.0428	2.3569	1.968	1.3377
138	0.9688	0.0196	0.0196	1.3145	1.2465	0.7132
139	0.5605	0.0039	0.0039	0.9244	0.8545	0.7766
140	6.2342	0.0814	0.0814	2.307	2.0258	1.6685
141	0.7422	0.0163	0.0163	1.1791	0.9129	0.7987
142	2.6725	0.0641	0.0641	2.0037	1.4645	1.2452
143	4.4352	0.055	0.055	2.324	1.8664	1.1495
144	3.601	0.047	0.047	1.9779	1.697	1.3897
145	1.2308	0.0427	0.0427	1.665	1.0524	0.8376
146	0.5652	0.0158	0.0158	1.2376	0.8075	0.6291
147	5.4715	0.0149	0.0149	2.0819	1.9383	1.4805
148	0.1373	0.0034	0.0034	0.7494	0.4766	0.4128
149	8.4827	0.4123	0.4123	3.9905	3.3181	1.8273
150	6.1244	0.0317	0.0317	2.2486	1.7515	1.6605
151	0.2037	0.0088	0.0088	0.8597	0.575	0.4779
152	2.2619	0.0559	0.0559	2.1697	1.4886	0.9016
153	1.9141	0.0292	0.0292	1.5901	1.3765	0.9466

Appendix Table 12 Historical Database Treatment Plan Table

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1	1	14	9.7	5.5	12.6	1	125
2	1	8	10.7	4.5	12.2	0.5	125
3	1	8	10.5	5	11.5	0.5	125
4	1	8	9.8	4.4	12.2	0.4	125
5	1	8	10.4	4.8	13	0.3	125
6	1	8	9.9	5.4	11.6	0.3	125
7	2	8	9.7	3.5	8.3	1	110
8	2	8	9.5	4.4	7.9	0.6	110
9	2	8	8.9	3.7	7.6	0.4	110
10	2	8	9.8	3.8	7.3	1	110
11	2	8	8.3	4	7.4	0.6	110
12	2	8	8.7	4.4	7.2	0.4	110
13	2	8	9.7	4.7	6.9	0.6	90
14	2	8	10	4.8	6.2	1	90
15	2	8	9.1	4	6.7	0.3	90
16	2	8	9.2	4.5	6.4	0.3	90
17	2	8	10	5.5	6	0.4	90
18	3	18	7.1	8.3	8.9	1	90
19	3	18	7.7	7.7	7.7	1	90
20	3	8	7.2	8.6	7.5	0.5	90
21	3	8	8.5	8	8.8	0.6	90
22	4	18	5.1	8.3	8.6	1	90
23	4	18	5.1	9.1	9.4	0.4	90
24	5	8	8	8.7	12.7	0.7	90
25	5	8	7.9	8.6	13.4	0.8	90
26	5	8	8.5	9.3	13.1	0.9	90
27	5	4	7.7	9.5	13.1	0.6	90
28	5	4	8.6	9.2	13.6	0.7	90
29	5	4	8.3	8.8	13.6	0.6	90
30	6	14	9	11.5	10.1	1	90
31	6	8	8.7	11.7	9.2	0.8	90
32	6	8	9.8	11	10.3	0.4	90
33	6	8	9.1	11.4	10.8	0.7	90
34	6	8	9.7	11	10.9	0.6	90
35	6	8	9.2	11.4	9.2	0.6	90
36	7	14	9.9	5.4	11.8	1	125
37	7	14	9.9	4	11.7	0.9	125
38	7	14	9.5	4.4	13	0.8	125

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
39	7	8	10.1	5.3	12.7	0.6	125
40	7	8	9.2	4.9	13.5	0.7	125
41	7	8	8.8	5.1	11.6	0.6	125
42	7	8	8.9	4	11.9	0.4	125
43	7	8	9.8	4.7	11.1	0.3	125
44	7	8	9.2	5.9	12.1	0.4	125
45	8	8	7.9	7.7	12.8	0.8	125
46	8	8	8.5	8.1	12.6	0.8	125
47	8	8	7.7	7.8	12.2	0.8	125
48	8	8	8.1	8.1	13.1	1	125
49	9	18	12.2	8.3	3.9	1	90
50	9	8	12.9	7.3	4.6	0.6	90
51	9	8	12.6	9.5	4.5	0.5	90
52	9	8	13.3	8.6	4.6	0.3	90
53	9	8	12.5	7.7	5.2	0.5	90
54	9	8	11.8	8.5	5.2	0.5	90
55	9	8	12.6	9.1	5.1	0.4	90
56	10	18	6.6	11	11.3	0.9	90
57	10	8	7.2	12.4	11.9	0.6	90
58	10	8	7.7	11.7	11.7	0.3	90
59	10	8	7.8	10.1	11.2	0.3	90
60	10	8	7.4	12.3	11.1	0.4	90
61	10	8	7.9	11.3	10.8	0.2	90
62	10	8	7.6	12.2	10.5	0.6	90
63	10	8	7.9	10.2	10.3	0.6	90
64	10	8	6.9	11.2	10.1	0.3	90
65	10	8	7.7	11.4	9.7	0.5	90
66	10	8	7.8	10.3	9.6	0.4	90
67	10	8	7.7	12	9.7	0.5	90
68	11	14	8.6	14.8	7.2	0.4	90
69	11	14	10.1	14	7.2	0.6	90
70	11	14	9.8	15.4	7.5	0.7	90
71	11	8	8.9	13.8	6.7	0.5	90
72	11	8	10.3	14.5	8	0.5	90
73	11	8	10.3	15.6	8	0.5	90
74	11	8	8.4	14.3	6.5	0.4	90
75	11	8	9.2	13.9	6.3	0.5	90
76	11	8	9.8	14.7	6.6	0.3	90
77	11	8	9.9	13.5	6.5	0.3	90
78	12	8	10	13	11.6	1	72
79	12	8	10.6	13.5	11.3	0.7	72
80	12	8	10.2	12.9	10.8	0.7	72

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
81	12	4	9.8	13.6	11.4	0.6	72
82	12	4	9.9	13.5	11	0.7	72
83	12	4	9.6	13.1	11.1	0.3	72
84	12	4	10.6	12.5	11.1	0.8	72
85	12	4	10	12.5	11.2	0.3	72
86	12	4	10.3	13.9	10.9	0.6	72
87	12	4	10.7	13.9	10.9	0.6	72
88	12	4	10.6	13.2	10.7	0.4	72
89	13	18	11.6	9	13.8	1	90
90	13	4	12.8	9.6	13.9	0.4	90
91	13	4	12.8	9.7	13.5	0.3	90
92	13	4	13.1	9.7	13.8	0.3	90
93	13	4	11.6	9.1	12.7	0.5	90
94	13	4	11.1	8.5	12.9	0.4	90
95	13	4	10.9	9.1	12.9	0.3	90
96	13	4	12.3	9.1	13	0.2	90
97	13	4	11.9	9.6	12.9	0.3	90
98	13	4	11.3	9.5	12.9	0.3	90
99	13	4	12.5	8.5	14.2	0.3	90
100	13	4	11.8	8.6	12.9	0.2	90
101	13	4	11.5	8.1	13.2	0.2	90
102	14	8	10.6	10.6	10.9	0.9	72
103	14	8	9.7	10.4	10.7	0.6	72
104	14	8	11.2	11.2	10.7	0.6	72
105	14	4	9.9	11.1	10.7	0.5	72
106	14	4	10.5	11.2	10.6	0.5	72
107	14	4	11.2	10.2	10.6	0.6	72
108	14	4	10.5	10	10.5	0.4	72
109	14	4	9.6	11.3	10.6	0.4	72
110	14	4	11.5	10.6	10.6	0.3	72
111	14	4	10.7	10.6	10.4	0.2	72
112	14	4	10	11.1	10.4	0.2	72
113	15	14	13.1	5.2	8.2	1	110
114	15	14	11.7	5.6	7.7	0.7	110
115	15	14	11.8	4.3	7.6	0.6	110
116	15	14	13.1	5.5	6.6	1	110
117	15	14	11.6	4.6	6.3	0.9	110
118	15	14	11.6	6.2	6.4	0.4	110
119	15	8	12.9	4.2	7.9	0.3	110
120	15	8	12.9	6.4	7.5	0.4	110
121	15	8	12.3	5.7	5.6	0.6	90
122	15	8	11.4	5.3	5.6	0.4	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
123	16	8	9.8	16.6	7.8	1	72
124	16	8	9.8	15.8	8.3	0.6	72
125	16	8	9.7	17.2	8.5	0.9	72
126	16	8	9.8	16.2	8.8	1	72
127	16	8	9.8	17	9.1	0.6	72
128	16	8	9.9	16.4	9.5	0.7	72
129	17	14	6.7	9.8	10.9	1	90
130	17	8	7.2	9.1	11.8	0.8	90
131	17	8	6.2	9.3	11.8	0.6	90
132	17	8	6.3	8.6	11.2	0.3	90
133	17	8	5.9	8.9	10.8	0.5	90
134	17	8	6.8	10.7	11.4	0.4	90
135	17	8	7.6	10.4	11.4	0.3	90
136	17	8	6.2	9.8	9.9	0.7	90
137	17	8	6.9	9.2	10	0.7	90
138	17	8	6.9	10	10	0.4	90
139	18	8	9.5	8.3	13.2	0.9	90
140	18	8	9.5	8.4	12.3	0.9	90
141	18	8	9.1	7.7	12.7	0.4	90
142	18	8	8.9	8.5	12	0.3	90
143	19	8	10.6	12.1	11.9	0.8	72
144	19	8	9.8	12	11.3	0.9	72
145	19	8	10.7	13	11.8	0.8	72
146	19	8	10.6	11.8	11	0.5	72
147	19	4	11.5	11.6	11.1	0.7	72
148	19	4	11.5	12.6	11.1	0.8	72
149	19	4	11	13	11.2	0.6	72
150	19	4	10.2	12.7	11.3	0.6	72
151	19	4	9.9	13.2	11.5	0.6	72
152	19	4	9.3	12.6	11.3	0.5	72
153	19	4	11.9	12.2	11	0.6	72
154	19	4	10.2	11.2	11.6	0.6	72
155	19	4	9.6	12.6	11.7	0.4	72
156	19	4	10.9	11.5	11.9	0.3	72
157	19	4	11.3	12.1	12	0.5	72
158	19	4	9.8	11.5	11.8	0.4	72
159	19	4	11.2	12.7	11.9	0.4	72
160	19	4	11	12.4	11	0.4	72
161	20	8	8.6	11.2	12	0.9	72
162	20	8	8.6	11.9	11.7	1	72
163	20	4	8.5	11.7	12.3	1	72
164	20	4	8	11.3	12	0.8	72

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
165	20	4	8.6	11.4	11.5	0.6	72
166	20	4	8.3	12	12.1	0.4	72
167	21	14	10.3	10.4	10.7	0.9	90
168	21	14	9.6	9	10.4	0.7	90
169	21	14	10.9	10.7	9.3	0.9	90
170	21	14	9.9	9.5	9	0.8	90
171	21	8	11	9.3	8.2	0.7	90
172	21	8	10.5	10.4	8.2	0.7	90
173	21	8	11.5	10.2	8.6	0.3	90
174	21	8	11.1	8.9	9	0.3	90
175	21	8	11.6	9.6	9.2	0.3	90
176	21	8	10.1	11.1	8.9	0.3	90
177	21	8	9.5	10.4	8.9	0.2	90
178	21	8	10.8	9	9.8	0.2	90
179	21	8	11.4	9.8	10.1	0.3	90
180	21	8	9.8	11.2	9.7	0.3	90
181	21	8	9.2	10.4	9.7	0.3	90
182	21	8	11.3	10.9	10.4	0.3	90
183	21	8	10.7	9	10.7	0.4	90
184	21	8	9.2	10.1	10.7	0.3	90
185	21	8	9.8	9.2	11.3	0.6	90
186	21	8	10.3	9.8	11.4	0.3	90
187	21	8	10	9.8	8.2	0.3	90
188	21	8	10.4	9.1	11.4	0.2	90
189	21	8	9.6	10	11.3	0.2	90
190	22	14	13.5	5.9	13	1	135
191	22	8	13.1	6.9	13.6	0.8	135
192	22	14	12.9	7	12	0.5	125
193	22	8	13.5	6	11.8	0.5	125
194	22	8	14	7.1	12.5	0.4	125
195	22	8	12.4	6.4	11.4	0.6	125
196	22	8	14	6.8	13.4	0.4	135
197	23	8	6.5	10	6.4	1	90
198	23	8	6.9	9.6	6.7	0.3	90
199	24	8	11.4	9.3	13	0.9	90
200	24	8	11.3	9.9	13.5	0.6	90
201	24	8	12	9.3	13.5	0.5	90
202	24	4	11.5	9.3	13.6	0.5	90
203	24	4	11.3	10	12.8	0.8	90
204	24	4	12	10.1	13.1	0.8	90
205	24	4	12.2	10.1	13.3	0.6	90
206	24	4	12.6	10.2	13.2	0.6	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
207	24	4	12.6	10.2	13	0.6	90
208	25	8	10.7	11.5	12	0.8	72
209	25	8	10.4	10.8	11.5	0.3	72
210	25	4	10.4	11	12.1	0.4	72
211	25	4	10.3	11.7	11.6	0.5	72
212	25	4	11.3	11.4	11.6	0.6	72
213	25	4	9.6	11.4	11.9	0.5	72
214	25	4	9.6	11.4	11.6	0.3	72
215	25	4	9.8	10.5	11.5	0.3	72
216	25	4	10.7	11.1	11.2	0.2	72
217	25	4	10.6	10.5	11.1	0.2	72
218	25	4	11.1	10.7	11.2	0.6	72
219	25	4	9.9	11.6	11.5	0.1	72
220	25	4	10.6	10.6	11.7	0.2	72
221	25	4	9.7	10.9	11.5	0.2	72
222	25	4	10.3	10.4	11.3	0.2	72
223	26	8	11.3	10.1	12.9	1	90
224	26	4	12	10.3	13.1	0.7	90
225	26	4	12.4	10.4	13	0.4	90
226	26	4	11.2	9.9	13.4	0.3	90
227	26	4	11.6	9.8	13.4	0.3	90
228	27	4	9.4	14.6	8.9	1.1	90
229	27	4	8.9	14.7	8.7	0.6	90
230	27	4	9.5	14.6	8.4	0.8	90
231	27	4	9.2	15.1	8.3	0.8	90
232	27	4	9	14.6	8.3	0.6	90
233	27	4	9.2	15.1	8.1	0.4	90
234	28	18	9	11.1	6	1	90
235	28	8	8.6	11.7	6.9	0.6	90
236	28	8	9	10.2	6.9	0.8	90
237	28	8	8.8	10.9	7.1	0.4	90
238	28	8	9.2	9.8	6.2	0.7	90
239	28	8	9	12.1	6.1	0.4	90
240	28	8	8.3	11.8	6.3	0.2	90
241	29	18	11.3	10.1	8.6	1	90
242	29	8	10.3	10.3	7.9	0.5	90
243	29	8	11.7	9.6	9.5	0.4	90
244	29	8	10.4	10.3	9.2	0.3	90
245	30	8	12.2	11.6	10.3	0.8	90
246	30	8	12.9	10.9	10.7	0.7	90
247	30	8	12.1	10.3	10.6	0.6	90
248	30	8	11.5	11.1	10.9	0.6	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
249	30	8	12.2	11.6	11.2	0.6	90
250	30	8	13	10.1	11.8	0.5	90
251	30	8	11.6	12.1	11.9	0.6	90
252	30	8	12.5	11.2	12	0.6	90
253	30	8	11.7	10.4	12	0.5	90
254	30	8	11.2	11.3	11.7	0.3	90
255	30	8	13.5	10.9	11.8	0.3	90
256	30	8	13	10.1	12.5	0.6	90
257	30	8	11.9	11.1	12.4	0.4	90
258	31	14	13	13.2	10.6	0.9	90
259	31	14	13.3	15.2	9.7	0.6	90
260	31	14	12.5	13.4	9.3	0.4	90
261	31	14	12	15.4	9.6	0.4	90
262	31	8	13.9	13.6	9.6	0.7	90
263	31	8	14.1	12.8	10.4	0.3	90
264	31	8	14.1	14	10.3	0.3	90
265	31	8	12.9	14.5	9.2	0.3	90
266	31	8	11.7	14.3	9.2	0.4	90
267	31	8	13.4	12.9	9.7	0.2	90
268	32	18	14.4	8.8	8.4	1	90
269	32	8	14.1	8.2	7.6	0.7	90
270	32	8	13.3	8.6	7.9	0.7	90
271	32	8	13.7	8.6	9.1	0.4	90
272	32	8	14.9	9.2	9.1	0.8	90
273	33	18	9.9	9.3	13.1	0.9	90
274	33	8	9	9.2	12.3	0.3	90
275	33	8	9.8	10.4	12.6	0.3	90
276	33	8	9.1	10	12.3	0.3	90
277	33	8	10.2	9.5	14.2	0.4	90
278	33	8	10.5	10	13.6	0.2	90
279	34	18	13.9	4.2	7.2	1	110
280	34	8	13.9	4.3	8.6	0.8	110
281	34	8	13.7	5.2	8.2	0.6	110
282	34	8	12.6	4	7.1	0.6	110
283	34	8	12.8	4.4	8.2	0.4	110
284	34	8	14.8	4.9	8	0.4	110
285	34	8	13.6	3.5	8.2	0.3	110
286	34	8	14.7	4	8.1	0.3	110
287	34	8	13.4	5.4	7.4	0.3	110
288	34	8	14.4	5.3	7.6	0.2	110
289	34	8	12.8	4.7	7.5	0.2	110
290	35	14	10.5	8.3	10.4	1	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
291	35	14	10.8	8.4	12	0.6	90
292	35	8	10.4	9.6	10.4	0.6	90
293	35	8	11.8	8	10.6	0.5	90
294	35	8	9.9	9.3	11.2	0.6	90
295	35	8	11.7	7.9	11.4	0.6	90
296	35	8	9.9	8.3	11.5	0.3	90
297	35	8	10.1	9.1	11.8	0.3	90
298	35	8	11.6	8.3	12.2	0.4	90
299	35	8	10.7	7.6	11.3	0.2	90
300	35	8	11.9	9	11.2	0.5	90
301	35	8	9.7	9.2	10.4	0.3	90
302	35	8	10.3	9	9.7	0.3	90
303	36	14	13.3	5.7	12	1	125
304	36	14	11.5	4.8	11.7	1	125
305	36	14	9.6	4.9	11.2	0.6	125
306	36	14	12.2	6.4	11.4	0.6	125
307	36	8	11.6	7.4	10.9	1	125
308	36	8	11	6.3	10.9	0.7	125
309	36	8	8.4	4.7	11.2	0.9	125
310	36	8	12.9	4.7	11.4	0.6	125
311	36	8	10.6	5.2	10.9	0.3	125
312	36	8	12.4	5.3	12.7	0.7	125
313	36	8	13.6	6.4	12.6	0.6	125
314	36	8	11.5	4.9	12.9	0.6	125
315	36	8	13.3	5.6	13.2	0.6	135
316	36	8	12.5	5.4	13.7	1	135
317	36	8	11.4	5.1	13.5	0.7	135
318	36	8	13.5	6.1	13.6	1	135
319	36	8	12.7	7.5	11.3	0.4	125
320	36	8	9	5.8	10.9	0.4	125
321	37	14	8.5	9.6	10.7	1	90
322	37	8	8.5	9.6	10.7	0.6	90
323	37	8	7.6	10	11.1	0.9	90
324	38	8	9	9.8	13	0.8	90
325	38	4	8.5	9.8	12.5	0.3	90
326	38	4	8.8	9.8	13.5	0.3	90
327	38	4	8.3	9.5	13.3	0.4	90
328	38	4	8.3	10.2	12.8	0.4	90
329	38	4	8.3	9.5	12.8	0.3	90
330	38	4	7.9	10.3	12.9	0.6	90
331	38	4	9	10.3	12.6	0.3	90
332	38	4	9.2	9.7	12.5	0.2	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
333	39	8	12.2	5.7	11.7	0.8	110
334	39	8	11.7	4.8	11.9	0.7	110
335	39	8	11.1	5.5	12.1	0.8	110
336	39	8	12.5	3.9	11.2	0.8	110
337	39	8	11.4	4.2	11.3	0.5	125
338	39	8	11.4	5.2	11.3	0.4	110
339	39	8	11.8	6.2	11	0.6	110
340	39	8	12.6	5.3	10.9	0.5	110
341	39	8	11.9	2.9	11.5	0.6	125
342	39	8	12.8	5.7	11.5	0.3	110
343	39	8	11.9	4.6	10.9	0.4	110
344	39	8	12.8	5.7	11.6	0.3	110
345	39	8	12.8	5.7	11.6	0.3	110
346	39	8	12.8	5.7	11.6	0.3	110
347	39	8	12.8	5.7	11.6	0.3	110
348	40	14	11.3	3.1	11.3	1	110
349	40	8	11.2	3.1	10.1	0.8	110
350	40	8	11.5	4.5	9.4	0.7	110
351	40	8	11.4	3.5	9.6	0.5	110
352	40	8	10.3	2.5	11.3	0.4	110
353	40	8	11.1	4.4	11.5	0.6	110
354	40	8	10.3	3.6	11.5	0.4	110
355	40	8	10.7	2.7	10.4	0.2	110
356	40	8	10.7	3.4	9.7	0.2	110
357	40	8	12	5.7	12.7	0.9	110
358	40	8	11.5	4.8	12.7	0.8	110
359	40	8	11	5.5	12.9	0.6	110
360	40	8	12.1	4	12.3	1.1	110
361	40	8	12.3	2.9	12.1	0.9	125
362	40	8	11.4	5.1	12	0.5	110
363	40	8	12.2	2.3	12.4	0.3	125
364	40	8	11	4.2	12.1	0.5	110
365	41	14	12.3	10.7	11.3	1	90
366	41	8	11.5	10.2	10.5	0.5	90
367	42	14	10.2	6.5	5.3	1	90
368	42	14	9.9	5.5	6.6	1	125
369	42	14	9.8	7.3	6.5	1.1	125
370	42	14	11.3	6	6.2	0.8	125
371	42	14	10.9	6.1	7.7	1.2	125
372	42	8	9.6	6.3	7.6	0.8	125
373	42	8	9.8	7.3	7.6	1.1	125
374	42	8	11	7.2	6.7	0.5	125

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
375	42	8	11.7	6.6	6.7	0.5	125
376	42	8	10.2	4.9	7.2	0.6	125
377	42	8	11.3	4.9	6.9	0.6	125
378	42	8	10.9	7.6	6.1	0.5	125
379	42	8	9.2	6.4	5.7	0.5	125
380	42	8	9.5	7.5	5.5	0.5	125
381	42	8	10.4	5.4	5.8	0.4	90
382	42	8	11.3	6.2	5.4	0.6	90
383	42	8	9.6	5.4	7.4	0.4	125
384	42	8	10.9	7.2	5.5	0.3	125
385	42	8	9.2	6.3	6.8	0.4	125
386	43	14	6	8.8	11.7	1	72
387	43	8	5.6	10.2	11	0.9	90
388	43	8	5.7	10	11.8	0.4	90
389	43	8	5.2	9.4	11	0.3	85
390	43	8	6.4	9.6	12.2	0.2	90
391	44	14	11.9	9.6	12.1	1	90
392	44	8	12	9.3	12.5	0.9	90
393	44	4	12.6	9.2	11.8	0.8	90
394	44	4	12.7	10	12.2	0.8	90
395	44	4	13.1	10.1	12.4	0.9	90
396	45	18	8.8	5.8	10.4	1	110
397	45	18	10.5	5.9	9.5	0.6	110
398	45	18	9.2	6.2	8.2	0.8	110
399	45	8	9.5	4.6	9.8	0.4	110
400	45	8	8.6	5.1	9.1	0.7	110
401	45	8	9.5	6.9	10.2	0.5	110
402	45	8	9.5	7.1	9.2	0.5	110
403	45	8	8.4	6.7	9.2	0.3	110
404	45	8	7.7	5.8	9.5	0.3	110
405	45	8	9.7	4.7	8.7	0.5	110
406	45	8	8.1	5.7	8.7	0.2	110
407	45	8	8.3	6.9	10.2	0.2	110
408	46	18	5	9.4	8.4	1	90
409	46	8	5.9	9	9.4	0.9	90
410	46	8	5	11.1	8	0.5	90
411	46	8	5	11	8.8	0.7	90
412	46	8	4.6	10.4	7.6	0.3	90
413	46	8	5.9	10.6	8	0.2	90
414	46	8	5.1	8.6	7.7	0.3	90
415	47	8	9.1	2.7	9.3	0.8	110
416	47	8	9.2	3.3	8.5	0.8	110

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
417	47	8	8.5	3	8	0.6	110
418	47	8	9.5	3.7	7.7	0.8	110
419	47	8	9.2	4.2	8.3	0.4	110
420	47	8	9.5	3	7.8	0.3	110
421	47	8	8.9	3.3	7.3	0.6	110
422	47	8	9.6	3.7	6.7	0.6	110
423	47	8	9.6	4.4	7	0.4	90
424	47	8	9.6	4.5	6.3	0.6	90
425	47	8	9	3.7	6.3	0.5	110
426	47	8	9	2.7	8.7	0.2	110
427	48	18	12.7	12.3	7.3	1.5	90
428	48	4	13.9	12.1	6.7	0.5	90
429	48	4	13.6	11.4	6.7	0.6	90
430	48	4	11.9	11.6	6.6	0.4	90
431	48	4	13	11.7	6.3	0.6	90
432	48	4	13.4	12.2	6.3	0.2	90
433	48	4	12.3	12.1	6.3	0.3	90
434	49	14	10.2	13.2	10.2	1	72
435	50	8	11.9	8	11.5	0.9	90
436	50	8	11.5	9	11.4	0.5	90
437	50	8	11.3	7.7	10.7	0.6	90
438	50	8	11.8	8.3	10.7	0.4	90
439	50	8	11.6	7.9	10.2	0.4	90
440	50	8	11.9	8.9	10.3	0.5	90
441	50	8	10.8	8.5	10.4	0.4	90
442	51	18	9.6	6.1	8.2	1.8	90
443	51	8	9.4	6	9.4	0.7	90
444	51	8	10.2	6.4	9.2	0.4	90
445	51	8	8.9	6.4	9.2	0.4	90
446	51	8	8.6	6.2	8.3	0.3	90
447	51	8	9.2	5.4	9	0.3	90
448	52	8	6.6	12.9	7.6	0.9	90
449	52	8	6.9	12.7	6.7	0.9	90
450	52	8	6.1	11.8	7.5	0.9	90
451	52	8	6.3	11.5	6.7	0.9	90
452	52	8	6.8	12.2	6.1	1	90
453	52	8	6.7	13.8	7.5	0.8	90
454	52	8	6.9	13.6	6.7	0.5	90
455	52	8	6.5	11.4	6.2	0.3	90
456	52	8	7.1	13	6.2	0.3	90
457	53	14	10.3	6.5	5.5	0.8	90
458	53	14	11.2	7.9	6.8	0.6	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
459	53	14	11.3	7.5	5.6	0.6	90
460	53	8	10.3	6.9	4.4	0.6	90
461	53	8	10.2	7.6	4.7	0.5	90
462	53	8	10.3	6.1	4.7	0.4	90
463	53	8	11.1	6.7	4.7	0.4	90
464	53	8	10.2	7.7	5.7	0.5	90
465	53	8	11	8.4	6	0.3	90
466	53	8	10.6	6.7	6.5	0.5	90
467	53	8	9.9	8	6.9	0.7	90
468	53	8	11.1	7.9	7.3	0.3	90
469	53	8	10.7	8	5.1	0.3	90
470	53	8	11.9	7.9	6.2	0.3	90
471	53	8	11.6	7.1	6.6	0.3	90
472	53	8	10	8	6.3	0.2	90
473	53	8	10.6	8.6	6.6	0.2	90
474	54	18	11.2	2.9	10.1	1	120
475	54	18	11.8	4.7	9.8	0.9	120
476	54	18	11.4	3.6	8.3	0.7	120
477	54	8	10.4	4.3	10	0.6	120
478	54	8	10.3	3.2	9.2	0.5	120
479	54	8	10.3	4.4	9.1	0.7	120
480	54	8	12.2	3.1	9.2	0.4	125
481	54	8	12.5	4.3	8.7	0.6	125
482	54	8	10.9	2.7	8.8	0.4	120
483	54	8	11.7	5	8.5	0.4	125
484	54	8	11.6	3.8	10.7	0.3	120
485	55	18	8.3	8.1	10.8	0.9	110
486	55	18	6.1	7.3	10.7	0.9	110
487	55	18	8.7	6	10.2	0.4	110
488	55	18	9.5	6.9	7.7	0.8	110
489	55	8	7.3	7.4	11.5	0.3	110
490	55	18	8.9	5.4	8.8	0.4	110
491	55	18	6.7	6.7	9.1	0.6	110
492	55	18	8.9	7.6	9.2	0.4	110
493	55	8	10.1	5.6	7.9	0.6	110
494	55	8	7.8	7.4	8	0.7	110
495	55	8	7.6	5.8	8.3	0.8	110
496	55	8	6.7	7.7	8.3	0.4	110
497	55	8	9.9	5.9	9.3	0.5	110
498	55	8	7.5	8.2	9.1	0.3	110
499	55	8	6.6	8	9.5	0.3	110
500	55	8	5.9	5.9	10.6	0.5	110

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
501	55	8	8.3	6.9	11.3	0.2	110
502	55	8	9.2	7	10.7	0.2	110
503	55	8	9.9	7.1	9.6	0.3	110
504	55	8	5.9	5.9	9.7	0.3	110
505	55	8	7.3	5.5	8.9	0.2	110
506	56	14	9.1	9.5	12.4	1	90
507	56	14	8.7	9.7	11.2	1	90
508	56	8	8.1	9	11.2	1	90
509	56	8	8.9	8.6	11.3	1	90
510	56	8	9.5	9.6	11.2	0.9	90
511	56	8	8.3	8.6	11.8	1	90
512	56	8	9.5	9.8	12.4	1	90
513	56	8	8.6	9.1	12.5	1	90
514	56	8	7.7	10	12	0.8	90
515	56	8	9.1	10.2	11.7	0.4	90
516	56	8	9.8	9.6	11.7	0.4	90
517	56	8	9.1	9.3	12.9	0.2	90
518	57	14	10	11.8	12	0.8	72
519	57	8	11.1	12.9	12	0.6	72
520	57	8	11.3	11.4	12.3	0.6	72
521	57	8	11	12.9	12.6	0.5	72
522	57	8	11.2	11.5	11.6	0.4	72
523	57	4	9.9	12.3	11.2	0.5	72
524	57	4	10.6	12.1	11.2	0.3	72
525	57	4	10.6	11.6	11.1	0.4	72
526	57	4	9.9	11.8	11.1	0.5	72
527	57	4	10.5	12.5	11.3	0.3	90
528	57	4	10.4	13.2	11.8	0.5	72
529	57	4	9.8	12.8	11.7	0.3	72
530	57	4	10.6	11	12.2	0.6	72
531	57	4	10.7	11.2	12.5	0.3	72
532	57	4	11.6	12.3	11.6	0.4	72
533	57	4	10	12.5	12.6	0.6	72
534	57	4	11.6	13.3	12	0.4	72
535	57	4	9.9	12.7	12.1	0.2	72
536	58	8	9.1	9.2	11.6	0.9	90
537	58	4	9	8.9	12.2	0.6	90
538	58	4	8.5	9.2	12	0.5	90
539	58	4	8.8	8.6	11.8	0.6	90
540	58	4	8.6	8.7	12.1	0.3	90
541	58	4	9.2	9.2	12.1	0.3	90
542	59	8	8.8	9	13.1	0.9	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
543	59	8	8.5	10	13.5	0.6	90
544	59	8	8	8.8	13.8	0.7	90
545	59	8	8	9.7	14.2	0.9	90
546	59	4	7.6	9.6	13.8	0.4	90
547	59	4	9	9.3	13.9	0.4	90
548	59	4	7.9	10.2	13.8	0.3	90
549	59	4	8	9.6	13.2	0.3	90
550	59	4	8.7	9.7	13.1	0.2	90
551	59	4	8.9	8.7	13.7	0.3	90
552	60	18	6.5	9.4	11.5	1.1	90
553	60	18	6.5	7.1	11.3	0.6	125
554	60	18	7.7	6.7	9.5	0.7	110
555	60	18	6	7.1	9.7	0.7	110
556	60	8	6.6	8	12	0.7	125
557	60	8	7.7	8.6	10.6	1	90
558	60	8	5.6	8.6	10.5	1	90
559	60	8	7.2	9.6	10.3	0.8	90
560	60	8	5.9	9.8	10.1	1	90
561	60	8	7.6	8.3	9.5	1	90
562	60	8	6.3	8.9	9.7	0.2	90
563	60	8	8.2	8.3	9.1	0.4	90
564	60	8	8.3	7.5	10.3	0.3	90
565	60	8	8.6	6.5	10.2	0.2	110
566	61	18	7.2	8.4	12.3	1.5	125
567	61	8	7	6.9	12	0.8	125
568	61	8	7.8	7.3	11.6	0.6	125
569	61	8	6.3	7.5	11.6	0.4	125
570	61	8	6.9	7.7	11.1	0.8	125
571	61	8	7.8	8.1	11.1	0.6	125
572	61	8	6.2	8.4	11.3	0.4	125
573	61	8	7.4	8.9	11.2	0.4	110
574	61	8	6.2	7.6	11	0.4	125
575	61	8	7.1	7	11.4	0.3	125
576	62	8	6.5	10	6.4	1	90
577	62	8	6.9	9.6	6.7	0.3	90
578	63	8	7.8	9.1	11.4	1	90
579	63	4	7.3	9.5	11.5	0.6	90
580	63	4	7	9.5	11.4	0.6	90
581	63	4	7.5	9.5	11.1	0.3	90
582	63	4	8	9.3	10.8	0.7	90
583	64	18	13.4	13.7	11.5	1	80
584	64	18	13.5	11.8	12.1	0.7	70

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
585	64	18	13.5	14.5	9.9	0.7	50
586	64	14	12	12.8	11.8	0.7	70
587	64	8	13.9	13.2	11	0.8	50
588	64	8	12.2	13.9	10.7	0.9	70
589	64	8	14.3	13.1	10.1	0.8	70
590	64	8	14.1	13.7	9.2	0.5	70
591	64	8	12.4	14.2	10.1	0.5	70
592	64	8	13.6	15.7	9.9	0.5	50
593	64	8	13.5	15.2	11	0.9	50
594	64	8	14.4	12.4	11.4	0.5	70
595	65	4	8.8	9.6	12.6	1.1	90
596	65	4	8.3	10	12.5	1	90
597	65	4	8.9	9.9	12.3	1.2	90
598	65	4	8.8	9.5	12.2	1	90
599	65	4	8.3	9.9	12.1	0.8	90
600	65	4	7.9	10	12.2	0.9	90
601	65	4	8.9	9.8	12	0.5	90
602	66	4	10.4	8.8	11.4	1	90
603	66	4	10.7	8.9	11.1	0.7	90
604	66	4	10.3	8.8	11	1	90
605	66	4	11	9.2	11	0.3	90
606	66	4	10.9	8.6	10.7	0.5	90
607	66	4	10.6	9.1	10.7	0.9	90
608	66	4	10.9	8.9	10.5	0.6	90
609	66	4	11.2	9.5	10.9	0.6	90
610	66	4	10.5	8.6	10.6	0.6	90
611	66	4	11.2	9	11	0.3	90
612	67	18	13.3	8.8	8.7	1.3	90
613	67	8	14.1	9.9	8.3	0.7	90
614	67	8	12	8.9	9	0.9	90
615	67	8	12.2	8.3	8	0.8	90
616	67	8	12.2	9.4	8.2	0.6	90
617	67	8	13.6	9.8	7.7	0.7	90
618	67	8	12.1	8.6	7.4	0.8	90
619	67	8	12.7	8	7.3	0.9	90
620	67	8	13.6	8.6	7.4	0.9	90
621	67	8	12.7	8.3	6.8	0.9	90
622	67	8	13.2	9.4	7.3	0.7	90
623	68	8	9.9	10.1	9.6	0.7	90
624	68	8	9.9	9.7	8.8	0.5	90
625	68	8	10	10.4	8.8	0.4	90
626	68	4	9.8	10.6	9.3	0.3	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
627	68	4	9.7	10.2	8.5	0.3	90
628	68	4	10.4	10.1	9.2	0.2	90
629	69	18	14.9	5.7	7.1	1.5	90
630	69	14	13.7	4.5	6.6	1.1	90
631	69	14	14.9	5.9	5.9	0.9	90
632	69	8	13.8	5.5	7.4	0.8	90
633	69	8	13.5	5.8	6.8	0.7	90
634	69	8	13.7	5.6	6	0.8	90
635	69	8	14.7	4.7	6	0.7	90
636	69	8	13.8	4.8	5.7	1.1	90
637	69	8	14.6	4.5	7.2	0.6	110
638	69	8	14.9	6	5.7	0.3	90
639	69	8	14.3	6.6	6.6	0.3	90
640	69	8	14.9	4.5	6.5	0.3	110
641	70	18	8.5	12.2	9.8	0.8	65
642	70	8	9.1	13.6	10.2	0.5	65
643	70	8	9.7	12.9	10.3	0.5	65
644	70	8	9.7	12.7	10.9	0.3	65
645	70	8	9.1	13.1	10.9	0.4	65
646	70	8	8.8	13.8	10.7	0.5	65
647	71	14	9.7	12.4	11.2	1.4	72
648	71	4	9	11.8	11.6	1	72
649	71	4	9.7	11.6	11.6	1	72
650	71	4	9.5	13.4	11.1	0.9	90
651	71	4	8.8	12.5	11.7	1	90
652	71	4	9.2	13.3	11.7	1	90
653	71	4	10.3	13.3	11.2	1.1	90
654	71	4	9.8	13.4	11.7	1	90
655	71	4	10.4	13	11.8	1	90
656	71	4	10.5	12.3	11.7	1	90
657	71	4	10.3	11.8	11.6	1	72
658	71	4	9.5	11.7	12	0.6	72
659	71	4	10.3	11.8	11.9	0.5	72
660	71	4	8.9	12	11.9	0.5	72
661	71	4	9.4	13.5	12	1	90
662	71	4	9	13	11.9	0.7	90
663	71	4	9.1	12	11.2	0.7	72
664	71	4	10.7	12.2	12	0.6	72
665	71	4	10	13.3	12	0.8	90
666	72	14	10.4	7.5	5.3	1.5	90
667	72	8	10.6	6.3	5.7	1.3	90
668	72	8	11.2	8.6	5	0.9	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
669	72	8	10.3	8.5	4.7	0.6	90
670	72	8	9.6	7.8	4.6	0.7	90
671	72	8	11.6	7.8	4.8	1	90
672	72	8	11.5	8.9	4.6	0.7	90
673	72	8	10	8.8	4.3	0.9	90
674	72	8	10.7	8.1	4.3	0.6	90
675	72	8	10.7	8.9	4.3	0.5	90
676	72	8	11.5	6.9	4.9	0.5	90
677	72	8	10.3	8.6	5.3	0.4	90
678	73	8	9.2	8.3	12.9	1.1	90
679	73	8	9.2	8.5	13.5	1	90
680	73	4	9.8	8.3	13.3	0.8	90
681	73	8	9.4	8.4	14.1	0.8	90
682	73	4	9.9	8.4	12.7	0.8	90
683	73	4	10.2	8.3	14.2	0.5	90
684	73	4	9.3	8.5	14.9	1	90
685	73	4	9.2	9	14.9	1	90
686	73	4	9.3	9.2	14.4	0.8	90
687	73	4	9.1	9.2	14	1	90
688	73	4	9.6	8.9	14.1	0.6	90
689	73	4	10	8.3	14.6	0.8	90
690	73	4	10.3	8.1	13.8	0.5	90
691	73	4	10.1	8.6	14.3	0.3	90
692	73	4	9.3	8.6	14.6	0.8	90
693	73	4	10.2	8.2	13.4	0.9	90
694	73	4	9.9	8.3	13.8	0.7	90
695	74	18	12.2	10.2	7.1	1	90
696	74	8	12	10	8.4	1	90
697	74	8	12.8	9.5	6.4	0.5	90
698	74	8	11.7	9.6	8	0.4	90
699	74	8	11.8	10.7	7.8	0.4	90
700	75	14	6.5	7.5	7.2	1	110
701	75	8	8	6.6	7.7	0.8	110
702	75	8	6.5	8.8	8.3	1	110
703	75	8	7.5	7.6	8	0.7	110
704	75	8	6.2	6.2	6.9	0.6	110
705	75	8	7.4	6.2	6.8	0.6	110
706	75	8	7	6.6	7.9	0.6	110
707	75	8	7.6	7.1	7.2	0.6	110
708	75	8	6.7	8	6.6	0.6	110
709	75	8	7.2	6.7	6.6	0.6	110
710	75	8	6.8	8	8	0.6	110

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
711	75	8	6.7	5.8	7	0.4	110
712	75	8	6	7.9	7.7	0.3	110
713	76	4	12.2	10	13.3	1.1	90
714	76	4	12.6	10.3	13.2	1	90
715	76	4	13.1	10.4	13.2	0.8	90
716	76	4	11.9	10.3	13.1	1	90
717	76	4	12.3	9.9	13	1.1	90
718	76	4	12.1	10.1	12.7	1.3	90
719	76	4	12.2	10.4	13.3	0.4	90
720	77	8	6.5	5.5	10.7	2.2	125
721	77	8	6.8	6.6	10.8	2	125
722	77	8	7.2	7.6	10.7	1.6	125
723	77	8	5.9	6	10.9	1.5	125
724	77	8	7.4	5.5	10.2	1.3	125
725	77	8	6.6	4.7	10.1	1.2	125
726	77	8	7.4	6.5	10.3	1.1	125
727	77	8	7.9	5	9.5	1.5	125
728	77	8	7.7	6	9.6	1.5	125
729	77	8	8.3	7.2	9.7	1.2	125
730	77	8	7.2	4.8	9.8	0.7	125
731	77	8	7.4	6.9	6.7	1.5	110
732	77	8	6.4	5.5	6.8	1.2	110
733	77	8	5.9	6.3	7	1	110
734	77	8	8.2	5.8	7.4	1.3	110
735	77	8	7.6	5	7.7	1.3	110
736	77	8	7.5	7.2	7.5	1.2	110
737	77	8	7.6	6.4	8	0.9	110
738	77	8	7.5	7.4	8.2	1	110
739	77	8	7.9	5.6	8.4	1	110
740	77	8	6.9	4.3	8.2	1.2	110
741	77	8	7.6	4.9	8.4	0.4	110
742	77	8	7.4	8	8.5	0.9	110
743	77	8	7.1	8.5	8.7	1	110
744	77	8	7.3	4.7	8.7	1.2	110
745	77	8	7.7	6.7	8.8	1.1	110
746	77	8	6.6	9.2	8.6	1.1	110
747	77	8	5.9	9.5	8.7	1	110
748	77	8	5.1	9.3	8.6	1.2	110
749	77	8	7.7	5.5	9	0.7	110
750	77	8	7.3	6.7	6.3	1.2	110
751	77	8	6.9	6.1	6.5	1.2	110
752	77	8	6.9	6.9	7	1.2	110

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
753	77	8	7.7	5.7	7	1.3	110
754	78	14	9.2	11.2	5.3	1	90
755	78	8	9.3	10	5	0.6	90
756	78	8	8.6	11.9	5.1	0.3	90
757	79	18	6.6	9.6	10.3	1.5	90
758	79	8	6.7	8.2	9.6	0.9	110
759	79	8	5.5	8.5	9.9	0.8	110
760	79	8	5.9	7.7	10.3	0.8	125
761	79	8	6.1	8.2	11	1	125
762	79	8	6.1	10.3	11.2	0.8	110
763	79	8	7.1	8.2	10.5	0.4	110
764	79	8	5.6	8.6	10.7	0.6	125
765	79	8	6.2	7.8	11.9	1.5	125
766	79	8	6.1	9.9	11.7	0.9	110
767	79	8	6	9.2	11.3	0.5	110
768	80	14	9.6	14.4	9	1.5	72
769	80	8	10.2	13.2	9	1.1	72
770	80	8	9.7	13.1	9.7	0.8	72
771	80	8	10.6	13.4	9.7	0.8	72
772	80	4	10	13.2	10.1	0.7	72
773	80	4	9.3	13.3	10.1	0.6	72
774	80	4	10.2	13.9	10.1	0.8	72
775	80	4	10	14.6	9.9	1	72
776	80	4	9	14.1	9.9	0.8	72
777	80	4	9.5	14.1	9.9	0.3	72
778	80	4	8.8	14.6	9.7	0.6	72
779	80	8	8.9	14.5	8.6	0.3	72
780	80	4	10.3	12.8	10	0.6	72
781	81	14	11.7	9	12.4	1.5	90
782	81	14	11.3	10.1	11.4	1.5	90
783	81	8	12.1	8.6	11.5	1.1	90
784	81	8	11.1	9	11.5	0.9	90
785	81	8	11.8	8.9	10.9	1.5	90
786	81	8	12.1	10.1	12.1	0.6	90
787	81	8	11.3	10	12.4	0.5	90
788	81	8	11.9	9.8	10.7	0.7	90
789	81	8	12.7	8.9	11.8	0.3	90
790	82	8	9.2	10.6	15.1	1.1	70
791	82	8	10.1	10.7	14.9	1.2	70
792	82	8	9.2	9.9	14.6	0.8	70
793	82	8	8.6	10.3	14.5	1.1	70
794	82	8	9.5	10.7	14.4	0.8	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
795	82	8	10.2	10.9	14.2	1	90
796	82	8	9.2	10.2	13.9	0.8	90
797	82	8	9.5	10.7	13.6	1.5	90
798	82	8	9	9.7	14.1	0.7	70
799	82	8	8.7	10.1	13.5	0.7	90
800	83	8	9.2	10.6	15.1	1.1	70
801	83	8	10.1	10.7	14.9	1.2	70
802	83	8	9.2	9.9	14.6	0.8	70
803	83	8	8.6	10.3	14.5	1.1	70
804	83	8	9.5	10.7	14.4	0.8	90
805	83	8	10.2	10.9	14.2	1	90
806	83	8	9.2	10.2	13.9	0.8	90
807	83	8	9.5	10.7	13.6	1.5	90
808	83	8	9	9.7	14.1	0.7	70
809	83	8	8.7	10.1	13.5	0.7	90
810	84	8	8	9.5	11.5	1	90
811	84	8	8.6	9.6	12.3	1.1	90
812	84	8	8.6	10.3	11.9	0.6	90
813	84	4	7.7	10.3	12.1	1.2	90
814	84	4	7.7	10.1	12.6	1	90
815	84	4	8.6	9.4	11.8	0.6	90
816	84	4	8	9.1	11.9	0.4	90
817	85	8	9.2	1.9	10.2	1	125
818	85	8	9.5	2.2	11	1	125
819	85	8	9.3	2	11.8	1.4	125
820	85	8	10.3	1.8	11.5	1.2	125
821	85	8	9.6	2.4	10	1.1	125
822	85	8	8.8	2.2	10.5	1	125
823	85	8	8.8	2.1	11.3	1	125
824	85	8	9.9	2.2	10.7	1	125
825	85	8	9.8	2.5	11.4	0.5	125
826	86	14	9.7	4.3	8.2	1.1	110
827	86	8	10	7.4	7.6	1.1	110
828	86	8	9.3	5.8	7.7	1	110
829	86	8	10	8.6	6.8	1	110
830	86	8	10	7.6	6	0.9	110
831	86	8	9.6	4.7	7.3	0.7	110
832	86	8	9.8	5.7	7	0.6	110
833	86	8	9.9	8.5	6	0.8	110
834	86	8	9.1	5.3	8.2	0.3	110
835	86	8	9.6	6.7	7.6	1	110
836	86	8	9.9	7.5	6.8	0.7	110

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
837	86	8	9.8	6.6	6.8	0.7	110
838	87	14	10.5	14.2	10.7	1	90
839	88	8	10.4	11.7	4.7	1	90
840	88	8	10.4	10.6	4.7	0.9	90
841	88	8	10.6	12.7	5	0.6	90
842	88	8	11.9	12	4.7	0.6	90
843	88	8	11.3	13.4	5.2	0.8	90
844	88	8	10.5	14.2	5.5	0.7	90
845	88	8	10.6	11.5	5.5	0.7	90
846	88	8	10.5	12.4	5.8	0.3	90
847	88	8	10.5	13.2	5.8	0.3	90
848	88	8	10.4	11.3	6.2	0.4	90
849	88	8	10.5	10.5	6.2	0.6	90
850	88	8	10.4	9.5	6.1	0.7	90
851	88	8	10.6	8.6	6.1	0.6	90
852	88	8	10.5	13.7	6.3	0.5	90
853	88	8	10.4	12	6.5	0.3	90
854	88	8	10.5	12.7	6.7	0.5	90
855	88	8	10.6	12.7	7.2	0.7	90
856	88	8	10.4	11.5	6.9	0.4	90
857	88	8	10.4	10.6	6.9	0.6	90
858	88	8	10.5	9.6	6.5	0.3	90
859	88	8	10.5	14.5	6.4	0.6	90
860	88	8	10.9	10.4	5	0.3	90
861	88	8	11.5	11.5	4.6	0.3	90
862	88	8	11.5	14.1	5.4	0.4	90
863	88	8	10.4	11.9	7.1	0.2	90
864	89	8	12.4	9.5	12.2	1.1	90
865	89	8	12.6	8.9	11.7	1	90
866	89	8	11.8	9.9	11.8	0.7	90
867	89	8	11.7	9.2	11.4	1	90
868	89	8	11.4	10.4	11.3	0.8	90
869	89	8	11.5	9.5	10.8	0.7	90
870	89	8	12.3	8.8	11	0.7	90
871	89	4	11.4	10.5	11.9	0.8	90
872	89	4	11.9	9	12	0.6	90
873	89	4	13.1	9.8	12.1	0.8	90
874	89	4	11.9	9	10.6	0.8	90
875	89	4	11.3	10.4	10.7	1	90
876	89	4	11.3	9.8	10.5	0.8	90
877	89	4	11.3	9.3	10.2	0.8	90
878	89	4	11.7	10	12.2	0.3	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
879	89	4	12.2	9.8	11.1	0.3	90
880	90	8	9.5	8.6	9.8	1.2	90
881	90	4	9.2	7.8	10	0.8	90
882	90	4	9.9	7.9	9.7	0.6	90
883	90	4	8.9	8.7	9.4	0.6	90
884	90	4	9.9	8	9.2	0.5	90
885	90	4	9.2	7.8	9.4	0.5	90
886	90	4	9.1	8.3	9.1	0.6	90
887	90	4	8.7	8.1	8.8	0.7	90
888	90	4	8.3	8.5	8.9	0.4	90
889	90	4	9.3	8.6	8.7	0.5	90
890	90	4	8.9	8.6	8.8	0.5	90
891	90	4	9.7	9	8.6	0.5	90
892	90	4	9.9	9.4	8.6	0.5	90
893	90	4	9.9	9.9	8.5	0.5	90
894	90	4	9.9	10.2	8.3	0.6	90
895	90	4	8.9	8.5	10	0.4	90
896	90	4	8.5	8.9	9.1	0.7	90
897	90	4	9.5	8.3	9	0.4	90
898	90	4	9.4	9.2	8.6	0.6	90
899	90	4	9	7.7	9	0.3	90
900	91	14	9.5	8.6	7.5	1	90
901	91	14	8.8	6.9	7.7	0.8	90
902	91	14	8.8	7	6.4	0.8	90
903	91	14	9.7	8.6	6	1	90
904	91	14	9.5	9.7	5.2	0.8	90
905	91	14	8.8	5.5	7.7	0.8	90
906	91	8	9.2	7.8	8.3	0.8	90
907	91	8	9.4	9.5	8.1	0.6	90
908	91	8	9.3	8.6	8.7	0.7	90
909	91	8	8.8	6	8.7	0.8	90
910	91	8	9	7	9.1	1	90
911	91	8	9.1	8	9.4	0.9	90
912	91	8	9.3	9.7	8.9	0.6	90
913	91	8	8	8	7.5	0.8	90
914	91	8	7.9	6.1	7	0.7	90
915	91	8	8	8	6.6	0.7	90
916	91	8	9.7	10	6.2	0.7	90
917	91	8	9.7	9.9	7	0.9	90
918	91	8	9.5	6	6.8	0.6	90
919	91	8	8.5	8.3	5.3	0.6	90
920	91	8	10.2	8.5	5	0.6	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
921	91	8	9	7.2	9.7	0.5	90
922	91	8	9.8	11	7.1	0.6	90
923	91	8	9.6	10.6	7.9	0.6	90
924	91	8	10.1	6.7	6	0.7	90
925	91	8	9.9	11.1	5.2	1	90
926	91	8	9.8	12.1	5.1	0.9	90
927	91	8	9.9	13	5.2	0.7	90
928	91	8	10.6	13.7	5.4	0.6	90
929	91	8	9.7	14.1	5.5	0.6	90
930	91	8	8.9	14.5	5.5	0.5	90
931	91	8	8.3	14.3	5.3	0.4	90
932	91	8	9.9	12.2	5.7	0.3	90
933	91	8	11.3	13.9	5.3	0.8	90
934	91	8	10.4	14.6	5.9	0.9	90
935	91	8	10.4	15.8	6.5	1	90
936	91	8	11.2	15.2	6.2	0.7	90
937	91	8	9.7	15.4	6.1	0.7	90
938	91	8	8.7	15.7	6	0.6	90
939	91	8	8	7.5	5.7	0.5	90
940	91	8	8.4	9.7	4.8	0.5	90
941	91	8	8.9	13.8	5	0.3	90
942	91	8	10.1	13.4	5.9	0.3	90
943	92	14	10.1	11.9	8.2	1	72
944	92	8	10.2	13.1	8.6	0.6	72
945	92	8	10.1	13	9.3	0.7	72
946	92	8	10	12.3	9.5	0.7	72
947	92	8	9.2	12.2	9.1	0.7	72
948	92	4	9.8	11.6	9.2	0.6	72
949	92	4	10.3	11.6	9.2	0.5	72
950	92	4	9.7	12.5	9.8	0.7	72
951	92	4	10.4	12.2	9.7	0.5	72
952	92	4	10.7	11.7	9.3	0.6	72
953	92	8	10.3	12.7	7.9	0.3	72
954	92	8	10.8	12.3	8.7	0.3	72
955	92	4	9.4	12.7	8.3	0.8	72
956	92	4	10	11.1	8.3	0.7	72
957	92	4	9.2	11.7	9.3	0.3	72
958	93	8	12.9	12.4	7.7	1	72
959	93	8	12.4	12.7	6.8	0.7	72
960	93	8	12.3	12	7.3	0.7	72
961	93	8	12.8	13.2	7.5	0.6	72
962	93	8	12.1	12.8	7.4	0.5	72

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
963	93	8	12.7	11.8	6.9	0.5	72
964	93	8	13.2	12.5	7.1	0.5	72
965	93	8	12.2	12.1	6.7	0.4	72
966	94	4	10	8.8	11.5	0.8	90
967	94	4	10.1	9	10.7	0.4	90
968	94	4	10.5	9.1	10.4	0.5	90
969	94	4	10.3	8.9	11.1	0.8	90
970	94	4	9.9	8.9	11	0.4	90
971	94	4	10.4	8.8	11.5	0.4	90
972	94	4	10.5	8.9	10.7	0.3	90
973	94	4	10.1	8.6	11.3	0.2	90
974	95	8	10.5	9.3	9.5	0.9	90
975	95	8	10.3	8.8	9.6	0.8	90
976	95	8	9.8	9.2	9.5	0.6	90
977	96	18	13.8	7	8	1.4	110
978	96	8	14.9	7.6	8.4	0.6	110
979	96	8	13.4	5.7	7.1	0.7	110
980	96	8	14.5	6.1	7.1	0.6	110
981	96	8	12.9	7.5	8.6	0.3	110
982	96	8	14.1	5.7	8.6	0.5	110
983	96	8	14.7	6.7	8.8	0.3	110
984	97	14	12.6	10.1	10	1.3	90
985	97	8	12.5	10.8	10.7	0.6	90
986	97	8	12.3	11	10	0.4	90
987	97	8	12.5	10.8	9.4	0.6	90
988	98	18	12.2	7.2	11.4	1.6	125
989	98	8	11.1	6.2	10.9	1	125
990	98	8	11.2	6.4	11.7	0.7	125
991	98	8	13.3	7.9	11.2	0.7	125
992	98	8	13.6	6.8	10.7	0.8	125
993	98	8	12.4	6	10.7	0.6	125
994	98	8	13.1	6.2	10.5	0.7	125
995	98	8	11.3	7	10.4	0.9	125
996	98	8	11.3	5.7	10.3	1.1	125
997	98	8	12	6.6	10.1	0.7	125
998	98	8	12.2	5.6	10.3	1	125
999	98	8	13.3	7.7	10.7	0.6	125
1000	98	8	11.6	5.8	11.4	0.4	125
1001	98	8	10.7	6.3	10.3	0.5	125
1002	98	8	13.1	7.5	12	0.7	125
1003	99	14	8.6	4.8	9	1.6	125
1004	99	8	9.2	5.3	9.6	0.7	125

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1005	99	8	8	5.5	9.5	0.4	125
1006	99	8	8.7	5.7	9.8	1	125
1007	99	8	8.6	4.3	10	0.7	125
1008	99	8	8	4.9	10.3	1	125
1009	99	8	8.2	5.7	10.3	0.9	125
1010	99	8	9.1	5.2	10.2	0.6	125
1011	99	8	8.5	5.7	8.8	0.4	125
1012	99	8	8.6	4	8.8	0.5	125
1013	99	8	8.3	4.5	10.5	0.4	125
1014	99	8	9.3	6	9.7	0.4	125
1015	99	8	8	4.4	9.9	0.3	125
1016	100	18	6.7	9.3	7.6	1	90
1017	100	8	7.3	10	8.3	0.8	90
1018	100	8	7.4	10.1	7.5	0.4	90
1019	100	8	5.9	10.1	6.8	0.6	90
1020	100	8	5.9	8.9	6.8	0.5	90
1021	100	8	6.3	9.3	6.2	0.6	90
1022	100	8	6.9	9.3	6.6	0.3	90
1023	100	8	6.6	8.6	6.6	0.2	90
1024	101	14	13.3	5.7	12	1	125
1025	101	14	11.5	4.8	11.7	1	125
1026	101	14	9.6	4.9	11.2	0.6	125
1027	101	14	12.2	6.4	11.4	0.6	125
1028	101	8	11.6	7.4	10.9	1	125
1029	101	8	11	6.3	10.9	0.7	125
1030	101	8	8.4	4.7	11.2	0.9	125
1031	101	8	12.9	4.7	11.4	0.6	125
1032	101	8	10.6	5.2	10.9	0.3	125
1033	101	8	12.4	5.3	12.7	0.7	125
1034	101	8	13.6	6.4	12.6	0.6	125
1035	101	8	11.5	4.9	12.9	0.6	125
1036	101	8	13.3	5.6	13.2	0.6	135
1037	101	8	12.5	5.4	13.7	1	135
1038	101	8	11.4	5.1	13.5	0.7	135
1039	101	8	13.5	6.1	13.6	1	135
1040	101	8	12.7	7.5	11.3	0.4	125
1041	101	8	9	5.8	10.9	0.4	125
1042	102	18	8.5	12.2	9.8	0.8	65
1043	102	8	9.1	13.6	10.2	0.5	65
1044	102	8	9.7	12.9	10.3	0.5	65
1045	102	8	9.7	12.7	10.9	0.3	65
1046	102	8	9.1	13.1	10.9	0.4	65

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1047	102	8	8.8	13.8	10.7	0.5	65
1048	103	18	8.4	7.7	14.8	1.1	130
1049	103	18	8.8	8.4	13.1	0.6	70
1050	103	14	8.2	6.9	13.7	0.6	130
1051	103	14	9.5	7.3	13.6	0.5	130
1052	103	14	7.9	8.6	14	0.3	90
1053	103	14	9.1	8.5	14.5	0.4	70
1054	104	14	12.5	9.1	11.6	1	90
1055	104	8	12.1	10.1	11.2	0.7	90
1056	104	8	12.7	10.2	11.9	0.4	90
1057	104	8	11.8	9.9	11.8	0.4	90
1058	104	8	12.5	9	12.4	0.6	90
1059	104	8	13.1	10	12.2	0.3	90
1060	105	8	11.8	12.5	11.1	0.8	72
1061	105	8	12.6	12.6	11	0.9	72
1062	105	8	12.5	12.5	10.1	1	72
1063	105	8	11.5	12.3	10.5	0.4	72
1064	105	8	12.9	12.9	10.4	0.3	72
1065	105	8	12.3	12	10.7	0.2	72
1066	105	8	12.1	13	10.7	0.3	72
1067	106	4	7.5	10.3	11.9	1	90
1068	106	4	8.1	10	12	0.6	90
1069	106	4	8.6	10.1	11.9	1	90
1070	106	8	8	9.6	11.5	0.6	90
1071	106	4	8.8	10.2	11.6	0.8	90
1072	106	8	8.4	10	11	0.8	90
1073	106	4	7.7	10.4	11.4	1	90
1074	106	4	7.5	10.4	11.5	0.7	90
1075	106	4	8.3	10.4	11.2	0.7	90
1076	106	4	8.5	10.6	11.6	0.4	90
1077	106	4	7.5	10	11.4	0.5	90
1078	107	8	7	6.9	10.4	1	125
1079	107	8	7.6	6.1	10.3	1	125
1080	107	8	6.6	6.9	10.7	0.5	125
1081	107	8	7.3	5.7	10.9	0.6	125
1082	107	8	7	6	11.4	0.6	125
1083	107	8	7.6	6.9	10.3	0.5	125
1084	107	8	6.2	7.5	11	0.8	125
1085	107	8	6.6	6.6	11.7	0.7	125
1086	107	8	7.7	5.2	11.5	0.6	125
1087	108	8	8	6.8	6.3	1	90
1088	108	8	8.8	5.3	6.5	0.8	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1089	108	8	7.7	5.8	6.4	1	90
1090	108	8	7.1	6.5	6.4	1	90
1091	108	8	8.5	6.7	7	0.9	90
1092	108	8	7.1	5.9	7.1	1	90
1093	108	8	9.1	5.6	7	1	90
1094	108	8	7.6	6.6	7.3	0.8	90
1095	108	8	8	5.3	7	0.5	90
1096	108	8	8.6	6.4	7.5	0.6	90
1097	108	8	7.3	4.9	7.3	0.4	90
1098	108	8	8	5.8	7.6	0.4	90
1099	108	8	9.1	5.6	7.6	0.5	90
1100	109	4	7.5	9.9	13.2	0.9	90
1101	109	4	7	10.1	13.2	1.3	90
1102	109	4	7.6	9.8	12.9	1.1	90
1103	109	4	7.1	9.9	12.9	1	90
1104	110	14	13	8.3	7.2	1.2	90
1105	110	14	13.3	8.7	9.2	1	90
1106	110	8	12.6	9.3	7.7	0.9	90
1107	110	8	12.6	8.1	8.2	0.8	90
1108	110	8	13.9	7.9	7.9	0.6	90
1109	110	8	13.7	9.3	7.7	1.1	90
1110	110	8	13.7	9.3	8.8	0.9	90
1111	110	8	14.1	8.4	8.7	0.4	90
1112	110	8	13.3	7.7	8.6	0.5	90
1113	110	8	12.8	9.1	8.4	0.7	90
1114	110	8	14.1	8.6	7.6	0.5	90
1115	110	8	14	9.2	8.3	0.4	90
1116	110	8	13.1	8.9	9.5	0.1	90
1117	111	4	11.2	13.4	9.8	1	90
1118	111	4	11.1	14.1	9.7	0.7	90
1119	111	4	11	13.7	9.8	1.2	90
1120	111	4	11	13.3	9.5	1.1	90
1121	111	4	11.4	13.9	9.7	1.2	90
1122	111	4	11.4	13.5	9.4	1	90
1123	111	4	11.1	14	9.4	1	90
1124	111	4	11.1	13.6	9.2	0.7	90
1125	111	4	11.4	13.9	9.2	0.5	90
1126	112	8	13.8	14.8	8.8	1	72
1127	112	8	13.3	14.9	8.5	0.8	72
1128	113	14	8.8	7.9	13.3	0.9	90
1129	113	14	10.2	7.9	13.2	0.7	90
1130	113	14	8.4	7.7	12.1	0.4	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1131	113	14	8.6	7.4	11.3	0.6	90
1132	113	8	9.6	8.2	12.5	0.8	90
1133	113	8	8.6	7.2	12.2	1	90
1134	113	8	9.7	8.2	11.8	1	90
1135	113	8	9.9	7.6	12.4	0.7	90
1136	113	8	9.6	7.3	13.7	1	90
1137	113	8	9.5	8.3	13.5	0.7	90
1138	113	8	8.9	7.5	10.7	0.9	90
1139	113	8	8.1	7.9	10.7	0.8	90
1140	113	8	8.2	8	12.4	0.4	90
1141	113	8	8.4	8	11.4	0.4	90
1142	114	14	8.8	8.8	10.2	1	90
1143	114	14	7.9	10.3	11.2	1	90
1144	114	14	8.3	11.4	11.5	1	90
1145	114	14	7.9	11.5	12.6	1.1	90
1146	114	14	6.7	11.9	12	0.7	72
1147	114	14	8.5	8.9	11.2	1	90
1148	114	14	6.7	13.6	11.3	1	72
1149	114	14	6.6	13.4	10.6	0.6	72
1150	114	14	7.9	9.7	9.9	0.5	90
1151	114	14	7	8.6	11.1	0.3	90
1152	115	4	10	10.5	11.4	0.9	72
1153	115	4	9.6	10.6	11.7	1	72
1154	115	4	10.3	10.6	11.6	0.8	72
1155	115	4	9.9	10.9	11.7	1	72
1156	115	4	10	10.3	11.8	1	72
1157	115	4	9.3	10.7	11.9	1	72
1158	115	4	10.5	10.5	12	1	72
1159	115	4	10.5	10.9	11.7	0.5	72
1160	115	4	10.7	10.7	11.9	0.6	72
1161	115	4	10.3	11	12	0.5	72
1162	115	4	10.7	10.6	12.2	0.3	72
1163	116	8	9.6	11	10.9	0.9	90
1164	116	8	10.5	12	10.6	0.9	90
1165	116	8	10.7	11.1	10.7	0.9	90
1166	116	8	9.4	12	10.4	0.9	90
1167	116	8	10.5	12.5	10.1	1	90
1168	116	8	9.5	10.9	10.2	1	90
1169	116	8	9.6	11.9	10	0.5	90
1170	116	4	11.3	12.3	9.8	1	72
1171	116	4	10.7	12.6	9.7	1	72
1172	116	4	9.9	10.6	9.7	0.8	72

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1173	116	4	9.9	11.2	9.7	0.8	72
1174	116	4	9.2	12.2	9.5	0.9	72
1175	116	4	9.5	11.6	9.4	0.6	72
1176	116	4	10.3	11.3	9.9	1	72
1177	117	8	11.2	9.7	10.6	1.2	90
1178	117	8	11.2	10.9	10.7	1.2	90
1179	117	8	12	9.8	11.2	1.1	90
1180	117	8	11.5	10.7	11.3	1	90
1181	117	8	11.4	11.5	11.5	1	90
1182	117	8	11.5	10.3	11.8	1	90
1183	117	8	12	9.7	12	1	90
1184	117	8	11.6	10.8	12.2	1	90
1185	117	8	11.9	9.7	12.6	1	90
1186	117	8	11.5	10.3	12.8	1	90
1187	117	8	12.9	9.8	12.1	1	90
1188	117	8	12.7	10.7	12.3	0.7	90
1189	117	8	13	9.6	12.7	1	90
1190	117	8	11.2	10.2	10.6	0.6	90
1191	117	8	12.2	9.8	13	0.8	90
1192	117	8	13.1	9.5	13.2	0.8	90
1193	117	8	13.8	9.3	13.2	0.9	90
1194	118	8	14.8	9.3	10.3	0.6	110
1195	118	8	14.9	8.5	10.2	0.6	110
1196	118	8	14.6	9.2	9.9	1	110
1197	118	8	14.2	8.4	9.2	0.8	110
1198	118	8	15	7.9	9.6	1	110
1199	118	8	15	9	9.4	0.7	110
1200	118	8	14.1	7.3	9.5	0.8	110
1201	119	14	13.2	7.3	11.3	1.4	110
1202	119	8	13.6	8.5	11.4	1	110
1203	119	8	14	7.4	12	1	125
1204	119	8	12.9	8.3	12.2	1	110
1205	119	8	12.4	7.4	12.1	0.8	110
1206	119	8	13.4	7.4	12.5	1	125
1207	119	8	14.1	8.3	11.8	1	125
1208	119	8	13.2	7	12.2	0.7	125
1209	120	18	13.6	5.6	7.4	1.6	90
1210	120	14	13.5	4.4	8.3	1	110
1211	120	14	13.7	6.6	8.3	1	90
1212	120	14	13.2	4.8	9.2	1.2	110
1213	120	14	14	6.4	9.2	1.3	90
1214	120	8	12.7	5.7	8.3	1	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1215	120	8	14.1	5.1	9.4	1	110
1216	120	8	13.7	6.7	7.4	0.5	90
1217	120	8	12.7	6.4	7.6	0.5	90
1218	120	8	13	5.8	9.2	0.5	90
1219	121	8	8.2	10	11.8	0.9	90
1220	121	4	7.6	10	11.9	0.9	90
1221	121	4	7.7	9.9	11.6	0.8	90
1222	121	4	8.8	10	11.5	0.7	90
1223	121	4	8.3	9.5	11.5	0.5	90
1224	121	4	8.7	9.6	11.6	0.4	90
1225	122	8	10.5	9.7	7.3	1	90
1226	122	4	10.6	9.4	7.7	1	90
1227	122	4	10.9	8.8	7.9	1	90
1228	122	8	10.9	7.9	7.7	1	90
1229	122	8	10.3	8.8	7.3	1	90
1230	122	8	11	9	7	1	90
1231	122	8	11.2	8	6.9	1	90
1232	122	8	10.3	8.1	6.6	1	90
1233	122	8	9.9	9.5	6.7	0.8	90
1234	122	8	9.8	11	6.2	0.8	90
1235	122	8	12.3	10	6.1	0.8	90
1236	122	8	10.7	9.9	6.3	0.5	90
1237	122	8	11.7	8.3	6.2	0.6	90
1238	122	8	11.6	9.3	6.2	0.6	90
1239	122	8	10.4	10.6	6.4	0.6	90
1240	122	8	10.6	11.5	6.4	0.5	90
1241	122	8	10.2	12	6.3	0.5	90
1242	122	8	12.2	8.9	6.2	0.6	90
1243	122	8	11.8	10.7	6	0.6	90
1244	122	8	9.6	8.7	6.2	0.6	90
1245	122	8	10.7	8.7	6.1	1	90
1246	122	8	9.3	9.8	6.2	0.6	90
1247	122	8	9.5	11.8	6.1	0.6	90
1248	122	8	11.1	10.7	6	0.5	90
1249	122	8	9.2	12.4	6.2	0.5	90
1250	122	8	12.6	11	6.1	0.9	90
1251	122	8	8.8	10.6	6.1	0.7	90
1252	122	8	9	11.7	6.1	0.4	90
1253	122	8	9.4	13	6.1	0.5	90
1254	122	8	8.6	12.2	6	0.6	90
1255	123	14	11.1	5.9	11.8	1.3	125
1256	123	14	11.1	6.2	10.7	0.4	125

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1257	123	14	10.4	5.1	11.3	0.3	125
1258	123	14	10.5	6.4	12.6	0.3	125
1259	124	18	13.5	11.2	7.5	1	90
1260	124	18	11.3	10.9	6.6	1.1	90
1261	124	18	12.9	9.6	7.1	1	90
1262	124	14	13.5	10	8.6	1	90
1263	124	18	13.3	10.9	6.5	1	90
1264	124	14	12.6	10	8.1	0.8	90
1265	124	14	11.7	11.4	7.6	0.6	90
1266	125	18	7.3	6.2	10.5	1.4	110
1267	125	18	8.2	7.3	9.8	1.1	110
1268	125	18	8.2	4.3	9.8	1	110
1269	125	18	8.3	4.2	8.4	1.1	110
1270	125	18	7.9	6	8.4	0.9	110
1271	125	18	8.9	5.7	9.4	0.4	110
1272	125	8	8.2	4.7	7.8	1	110
1273	125	8	7	6.6	9.3	0.7	110
1274	125	8	7.8	7.2	8.9	1	110
1275	126	8	9.6	9	10.2	1	90
1276	126	8	9.9	8.8	11.2	1.1	90
1277	126	8	9.6	9.7	10.4	0.8	90
1278	126	8	9.6	9.6	11.8	0.8	90
1279	126	8	9.9	8.8	11.8	0.8	90
1280	126	4	8.6	9.9	11.3	1.1	90
1281	126	8	9.7	9.1	12.4	1	90
1282	126	4	9	9.2	11.7	1.1	90
1283	126	4	9	9.8	12.1	0.6	90
1284	126	8	9.8	8.9	10.6	0.7	90
1285	126	8	9.3	9.7	11.1	0.9	90
1286	126	4	9	9.1	10.7	0.5	90
1287	126	4	8.6	10	11	0.4	90
1288	126	4	10.2	8.9	9.9	0.4	90
1289	126	4	8.1	9.9	11.1	0.7	90
1290	126	4	8.9	9.1	11.3	0.6	90
1291	126	4	10	8.6	12.5	0.5	90
1292	127	8	12.7	13.3	8.5	1	90
1293	127	8	12.5	12.7	8.2	1.2	90
1294	127	8	13.3	12.7	8.7	0.8	90
1295	127	8	12.2	12.6	7.4	1	90
1296	128	8	12.4	10.6	11.8	1	90
1297	128	8	11.8	10.4	11.7	1	90
1298	128	8	12.2	9.9	11.4	0.7	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1299	128	8	12.4	10.1	11.4	1	90
1300	128	8	12.9	10	11.1	1	90
1301	128	8	12.3	9.6	10.7	1.1	90
1302	128	8	12.5	8.8	11.2	1.6	90
1303	128	8	13.5	9.4	11	1.2	90
1304	128	8	13.1	8.8	10.9	0.8	90
1305	128	8	12.9	8	10.7	1	90
1306	128	8	12	8.8	10.6	1.1	90
1307	128	8	12.7	9.3	10.7	1	90
1308	128	8	13.2	8.5	10.5	1	90
1309	128	8	12.1	9.4	10.6	0.6	90
1310	128	8	12.3	8.3	10.4	0.8	90
1311	128	8	12.7	8.7	10.4	0.8	90
1312	128	8	12.1	10	12.1	1.3	90
1313	129	8	14.7	9.2	10.1	1.1	90
1314	129	8	14.1	8.6	9.8	1	90
1315	129	8	14.9	9.7	9.9	1	90
1316	129	8	14.1	9.8	9.9	0.7	90
1317	129	8	13.5	10	9.3	1	90
1318	129	8	14.6	9.3	9.4	0.6	90
1319	129	8	13	10.6	9	1	90
1320	129	8	13.2	11.3	9	1	90
1321	129	8	12.3	11.5	8.6	1	90
1322	129	8	13.2	10	8.6	1	90
1323	129	8	13.8	10.9	8.4	1	90
1324	129	8	12.8	10.7	8.1	0.9	90
1325	129	8	12.5	11.8	8	1	90
1326	129	8	13.3	9.9	7.9	1	90
1327	129	8	13.8	10.6	7.7	1	90
1328	129	8	13.2	11.2	7.4	1	90
1329	129	8	12.9	10.5	6.6	1.4	90
1330	129	8	13.2	9.9	7.2	1	90
1331	129	8	12.9	10.9	7	0.7	90
1332	129	8	14.1	9.8	9.1	0.4	90
1333	130	18	12.2	10	10.8	1	90
1334	130	8	10.8	10	10.1	0.8	90
1335	130	8	11.9	10.8	10.3	1	90
1336	130	8	9.6	11.1	11	1	90
1337	130	8	12.1	11.3	11.1	1	90
1338	130	8	12	12	11.4	1	90
1339	130	8	12.1	8.2	11.2	0.7	90
1340	130	8	11.4	11.1	11.6	0.7	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1341	130	8	12.6	9.6	11.4	1	90
1342	130	8	11.4	9.5	11.8	1	90
1343	130	8	12.3	10.9	11.8	1	90
1344	130	8	12.6	8.4	11.9	1	90
1345	130	8	12.7	9.9	12	1	90
1346	130	8	9	9.9	11.6	1	90
1347	130	8	8.8	11.6	11.7	1	90
1348	130	8	12.3	11.1	12.7	1	90
1349	130	8	12.1	8.8	12.6	1	90
1350	130	8	10	9.8	12.6	1	90
1351	130	8	10.6	10	12.2	0.5	90
1352	130	8	11.2	9.4	12.5	1	90
1353	130	8	13	9.9	12.6	1	90
1354	130	8	12.1	11.2	12.3	1	90
1355	130	8	12.2	10.4	12.7	0.5	90
1356	130	8	8.9	10.7	11.6	0.6	90
1357	130	8	10.2	9.9	11.6	0.6	90
1358	130	8	10.7	9.8	11.2	1	90
1359	130	8	11.9	8.9	10.6	1	90
1360	130	8	10.5	11.1	11.2	1	90
1361	130	8	9.4	10.1	11.1	1	90
1362	130	8	10.4	10.7	10.6	0.4	90
1363	131	18	10.4	14	9.3	0.7	90
1364	131	18	10.6	13.6	10.5	0.8	90
1365	131	8	10.7	14.7	9.9	0.7	90
1366	131	8	11.3	13.6	9.7	0.7	90
1367	131	8	11.3	13.9	9.1	0.8	90
1368	131	8	11.1	14.8	9	0.8	90
1369	131	18	10.7	14.6	11.1	1.1	90
1370	131	8	10.6	13.8	9.1	0.6	90
1371	131	8	11.5	14.2	9.9	0.5	90
1372	132	4	10.9	10	11.6	1	90
1373	132	8	10.6	9.2	11.4	0.9	90
1374	132	8	9.9	9.2	11.3	0.9	90
1375	132	8	9.9	10	11.4	1	90
1376	132	8	10.1	9.9	10.7	1	90
1377	132	8	10.2	9.2	10.7	0.8	90
1378	132	8	10.2	9.1	11.8	1	90
1379	132	8	9.8	9.9	12	1	90
1380	132	4	10.4	9.6	12	0.8	90
1381	132	4	9.6	9.2	12.1	0.4	90
1382	132	4	10.6	9.3	10.6	0.8	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1383	132	4	10.5	9.7	10.5	0.4	90
1384	132	4	10.6	10.1	11.3	0.6	90
1385	132	4	10.9	9.5	10.9	0.8	90
1386	132	4	10.7	9	10.9	0.8	90
1387	132	4	9.9	10.2	12.1	0.2	90
1388	133	8	12.4	10.3	11.3	1	90
1389	133	8	11.3	10.3	11.5	1	90
1390	133	8	11.7	11.3	11.4	0.8	90
1391	133	8	12.6	9.8	12.1	1	90
1392	133	8	11.6	10	12.3	0.9	90
1393	133	8	13.1	10.7	12.6	0.6	90
1394	133	8	12.2	9.9	13	1	90
1395	133	8	11.7	10.4	13.1	1	90
1396	133	8	11.8	9.7	13.6	1.2	90
1397	133	8	11.7	11	12.4	0.8	90
1398	133	8	11.7	9.9	11.5	1	90
1399	133	8	13.3	10.2	11.9	0.5	90
1400	133	8	10.6	10.3	12	0.5	90
1401	133	8	13.2	10.1	12.6	0.4	90
1402	133	8	13.1	10	13	0.4	90
1403	133	8	12.4	11	12.7	0.3	90
1404	133	8	11.7	11.4	11.9	0.5	90
1405	133	8	12.7	10.8	11.9	0.3	90
1406	134	18	6.4	5.5	7.3	1.1	110
1407	134	8	5	5.6	7.3	1.1	110
1408	134	8	7.5	4.8	7.1	1.1	110
1409	134	8	5.6	6.2	7.7	1.2	110
1410	134	8	7.3	4.4	7.4	0.6	110
1411	134	8	6.4	4.4	7.7	0.8	110
1412	134	8	6.5	4.4	7	1	110
1413	134	8	6.5	6.2	7.3	0.8	110
1414	134	8	6.5	6.7	7.7	1	110
1415	134	8	7.3	5.4	7.5	0.9	110
1416	134	8	7.6	5.5	6.8	0.5	110
1417	134	8	5.6	4.7	7	1	110
1418	134	8	7.3	6.3	7.6	0.3	110
1419	134	8	6.9	4.5	6.5	0.5	110
1420	134	8	6.3	6.1	8.1	0.5	110
1421	134	8	7	4.8	8.1	0.4	110
1422	135	18	8.9	11.9	5.5	1	90
1423	135	18	9.7	13.8	5.7	0.8	90
1424	135	18	10.5	12.3	6.3	1	90

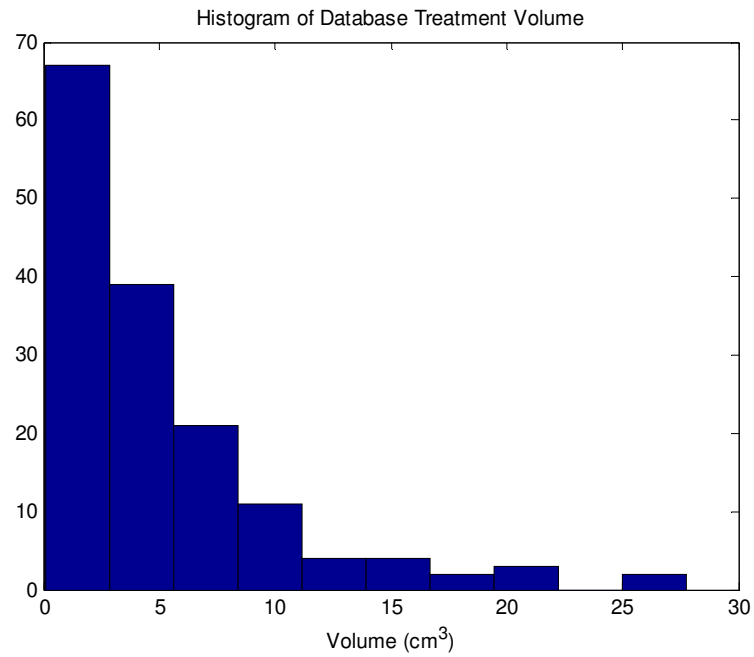
Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1425	135	18	7.8	12.6	6	0.6	90
1426	135	18	9.2	11.4	6.6	1	90
1427	135	18	8.9	13.6	6.6	0.7	90
1428	135	18	10.5	12.2	6.9	0.8	90
1429	135	18	7.5	11.8	7.4	0.8	90
1430	135	18	9.2	13.7	7.4	1	90
1431	135	18	9	11.7	7.6	0.8	90
1432	135	18	9.3	12.5	8.1	0.8	90
1433	135	14	9.7	13.2	4.9	0.6	90
1434	135	14	8.6	11.4	5	0.4	90
1435	135	14	9.4	12.5	4.9	0.3	90
1436	136	8	10.6	12.2	13.4	1.2	90
1437	136	8	9.8	12.1	13	1.2	90
1438	136	8	10.8	13	12.7	1	90
1439	136	8	10.7	12.2	12.7	1.1	90
1440	136	8	10.4	13.1	12.4	1	90
1441	136	8	10.9	12.3	12	0.5	90
1442	136	8	10.5	11.7	13.4	0.3	90
1443	137	14	11.2	8.4	4.8	1	90
1444	137	14	11.6	9.5	5.7	1	90
1445	137	14	10.7	8.2	5.7	1	90
1446	137	14	10.5	9.5	5.6	1	90
1447	137	14	10.6	8.8	6.5	0.4	90
1448	138	4	10.7	12.6	11.4	1	72
1449	138	4	11	12.2	11.4	1	72
1450	138	4	11.6	12.5	11.4	1.1	72
1451	138	4	10.7	12.9	11.2	0.6	72
1452	138	4	11.8	12.9	11.3	0.8	72
1453	138	4	10.9	13.4	11.2	1	72
1454	138	8	11.2	12.7	10.9	1.4	72
1455	138	4	11.5	13.4	11.2	0.8	72
1456	138	4	11.2	13.3	10.7	0.4	72
1457	138	4	11.4	12.2	11.6	0.3	72
1458	138	4	10.8	13.2	10.9	0.5	72
1459	139	8	8.4	10.2	5.1	1.6	90
1460	139	4	8.4	10.1	6	1.3	90
1461	139	4	8.6	9.8	5.6	1.1	90
1462	139	4	8.4	10.4	5.7	1	90
1463	139	4	8.1	10.1	5.6	1	90
1464	139	4	8.8	10.2	5.5	1	90
1465	139	4	8	9.8	5.3	0.6	90
1466	140	8	10.5	6.4	9.5	1.4	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1467	140	8	9.9	6.4	9	1	90
1468	140	8	10.1	5.1	9	0.5	125
1469	140	8	9.3	5.3	9	0.7	125
1470	140	8	10.7	5.4	9.6	1.1	125
1471	140	8	9.7	6.4	9.8	1	90
1472	140	8	11	7.1	9.7	0.9	90
1473	140	8	10.8	6	10.4	1	125
1474	140	8	9.8	6	10.5	1.4	125
1475	140	8	10	6.7	10.6	1.2	90
1476	140	8	11	7.2	10.2	1.3	90
1477	140	8	10.6	6.1	11	0.6	125
1478	140	8	9.2	6.9	8.9	0.6	90
1479	140	8	8.6	7.3	8.9	0.6	90
1480	140	8	10.6	7	9	1	90
1481	140	8	11.1	7.7	9	0.2	90
1482	140	8	10.5	5.8	9	0.7	125
1483	140	8	9.2	5.9	9.2	0.2	125
1484	140	8	9.8	5.4	9.7	0.5	125
1485	141	4	9.5	8.7	9.1	1.3	90
1486	141	4	10.1	8.8	8.9	1	90
1487	141	4	9.3	9	8.8	1	90
1488	141	4	9.2	8.6	8.7	1	90
1489	141	4	9.8	8.3	8.6	1	90
1490	141	4	9.5	9.2	8.5	1	90
1491	141	4	9.5	8.6	8.3	1	90
1492	141	4	9.5	9	8.2	1	90
1493	141	4	10.2	8.2	8.6	1	90
1494	141	4	10.4	8.8	8.6	1	90
1495	141	4	9.8	8.3	8.2	0.6	90
1496	142	8	12.1	10.5	12	1.3	110
1497	142	8	11.8	9.8	12.6	1	110
1498	142	8	12.2	10.8	12.9	1	110
1499	142	4	11.6	11	12.5	1.1	110
1500	142	4	11.2	9.9	13.1	1	110
1501	142	4	11.3	10.8	13.2	1	110
1502	142	8	12	10.5	13.6	1.1	110
1503	142	8	11.9	9.1	13.3	1	110
1504	142	4	11.8	9.6	14	1.2	110
1505	142	4	11.2	9.9	13.7	1.1	110
1506	142	4	11.3	10.3	12.5	0.4	110
1507	142	4	12.7	10.5	12.4	1	110
1508	142	4	12.5	9.8	13.5	0.7	110

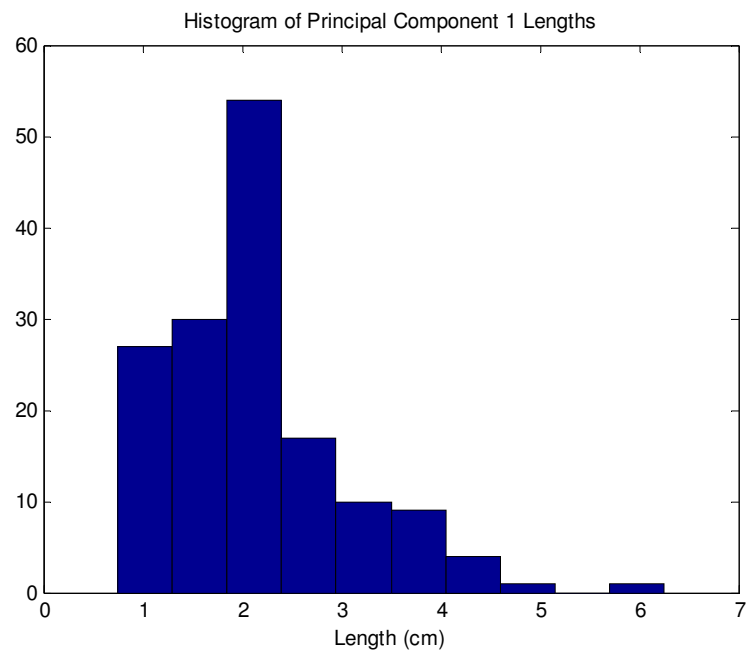
Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1509	142	4	11.2	10.8	12.9	0.4	110
1510	143	14	8	6.3	13.5	1.2	125
1511	143	14	7.1	7.4	12.6	1.1	125
1512	143	8	8.2	7.5	13.2	1	125
1513	143	8	7.3	5.8	12.9	1	125
1514	143	8	8	6.7	12.4	1	125
1515	143	8	7.3	6	12.4	0.9	125
1516	143	8	6.4	6.7	12.4	0.5	125
1517	143	8	8	7.4	12.2	0.6	125
1518	144	8	8.8	11.9	6.5	0.9	90
1519	144	8	9.5	12.4	6.7	1.1	90
1520	144	8	9.6	11.2	6.6	1	90
1521	144	8	10.2	12.1	6.6	1	90
1522	144	8	10.2	12.2	7.5	1.1	90
1523	144	8	9.3	12	7.7	1.1	90
1524	144	8	9.5	12.9	8	1	90
1525	144	8	10.2	12	7.9	0.7	90
1526	144	8	10.1	11.1	7.2	0.7	90
1527	144	8	8.9	11.9	7.1	0.5	90
1528	144	8	9.8	11.3	8	0.3	90
1529	144	8	9.2	11.2	7.3	0.3	90
1530	145	8	8	9.1	7.3	1.2	90
1531	145	4	7.7	9.1	8.1	1.1	90
1532	145	4	7.9	8.3	7.5	1.2	90
1533	145	8	7.7	9.9	8.2	0.8	90
1534	145	8	7.6	9.5	7.5	0.6	90
1535	145	4	7.8	8.6	8	0.9	90
1536	145	4	7.3	9.9	8.3	0.3	90
1537	145	4	8.4	8.3	7.4	0.6	90
1538	145	4	7.6	9.2	8.3	0.2	90
1539	145	4	7.1	9.2	8.2	0.4	90
1540	145	4	7.7	8.3	7.1	0.4	90
1541	146	8	11.9	8.5	11.9	1.6	90
1542	146	4	12.6	8.8	11.8	1	90
1543	146	4	12.2	8.3	11.5	0.4	90
1544	146	4	11.9	8.6	11.4	0.3	90
1545	147	18	13.3	10.2	11.9	1.8	90
1546	147	18	12.8	10.2	12.7	0.4	90
1547	147	8	12.3	9.6	12.4	0.9	90
1548	147	8	12.3	11	12.4	0.9	90
1549	147	8	12.2	10.9	11.5	0.9	90
1550	147	8	12.3	10	11.7	0.5	90

Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1551	147	8	12.9	11	11.3	0.8	90
1552	147	8	12.7	10	11	0.6	90
1553	148	4	8	11	12.5	1.2	90
1554	148	4	7.6	10.9	12.3	0.6	90
1555	148	4	7.7	11	12.7	0.6	90
1556	148	4	8	10.8	12.7	0.5	90
1557	149	8	6.5	5.5	10.7	2.2	125
1558	149	8	6.8	6.6	10.8	2	125
1559	149	8	7.2	7.6	10.7	1.6	125
1560	149	8	5.9	6	10.9	1.5	125
1561	149	8	7.4	5.5	10.2	1.3	125
1562	149	8	6.6	4.7	10.1	1.2	125
1563	149	8	7.4	6.5	10.3	1.1	125
1564	149	8	7.9	5	9.5	1.5	125
1565	149	8	7.7	6	9.6	1.5	125
1566	149	8	8.3	7.2	9.7	1.2	125
1567	149	8	7.2	4.8	9.8	0.7	125
1568	149	8	7.4	6.9	6.7	1.5	110
1569	149	8	6.4	5.5	6.8	1.2	110
1570	149	8	5.9	6.3	7	1	110
1571	149	8	8.2	5.8	7.4	1.3	110
1572	149	8	7.6	5	7.7	1.3	110
1573	149	8	7.5	7.2	7.5	1.2	110
1574	149	8	7.6	6.4	8	0.9	110
1575	149	8	7.5	7.4	8.2	1	110
1576	149	8	7.9	5.6	8.4	1	110
1577	149	8	6.9	4.3	8.2	1.2	110
1578	149	8	7.6	4.9	8.4	0.4	110
1579	149	8	7.4	8	8.5	0.9	110
1580	149	8	7.1	8.5	8.7	1	110
1581	149	8	7.3	4.7	8.7	1.2	110
1582	149	8	7.7	6.7	8.8	1.1	110
1583	149	8	6.6	9.2	8.6	1.1	110
1584	149	8	5.9	9.5	8.7	1	110
1585	149	8	5.1	9.3	8.6	1.2	110
1586	149	8	7.7	5.5	9	0.7	110
1587	149	8	7.3	6.7	6.3	1.2	110
1588	149	8	6.9	6.1	6.5	1.2	110
1589	149	8	6.9	6.9	7	1.2	110
1590	149	8	7.7	5.7	7	1.3	110
1591	150	14	7.4	7.3	10.9	1.3	125
1592	150	14	6.4	7.3	10.9	1.3	125

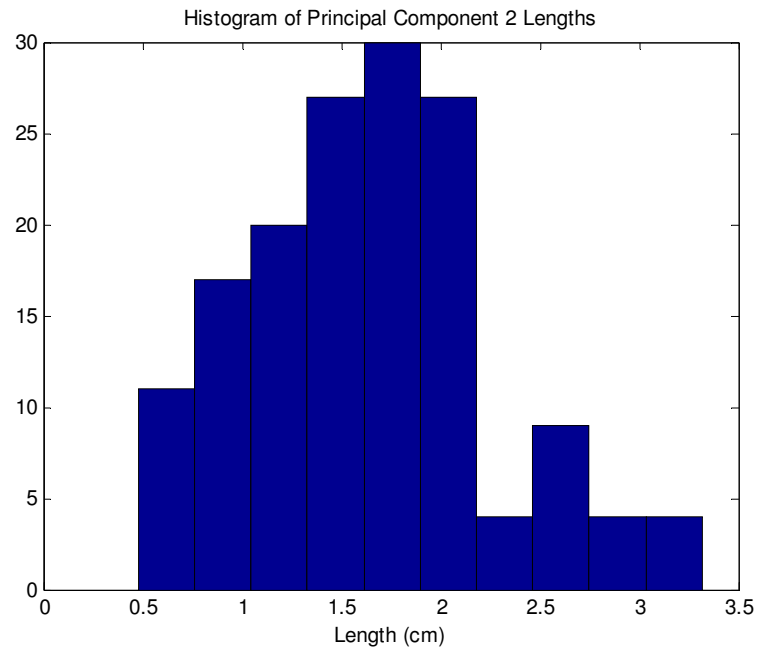
Isocenter Index	Structure Index	Collimator	X	Y	Z	Weight	γ Angle
1593	150	14	7.5	6.7	9.7	1.3	125
1594	150	14	6.6	7.4	9.7	0.7	125
1595	150	8	7.6	7.7	10	1	125
1596	150	8	6.4	6.6	9.8	1	125
1597	150	8	7.1	6.4	10.7	0.6	125
1598	150	8	7.5	7.6	9.4	0.3	125
1599	151	4	7.7	9.1	13.6	1.1	90
1600	151	4	7.2	9.4	13.5	1	90
1601	151	4	7.7	9.3	13.3	0.9	90
1602	151	4	7.3	9.5	13.2	0.8	90
1603	151	4	7.7	8.9	13.3	0.4	90
1604	152	8	8	8.9	11.6	1	90
1605	152	8	8.3	9.6	11.1	1.3	90
1606	152	8	8.5	8.6	11	1.1	90
1607	152	4	7.7	8.6	11.1	0.9	90
1608	152	4	8.2	8	10.7	0.9	90
1609	152	4	8.6	7.7	10.5	1.2	90
1610	152	4	8.8	8.3	10.5	1.1	90
1611	152	4	8.6	9.1	10.5	0.7	90
1612	152	4	8.8	8.6	10.1	1	90
1613	152	4	7.4	9	11.3	0.7	90
1614	152	4	8.6	9.5	10.6	0.3	90
1615	153	14	10	6.7	5.7	1.2	90
1616	153	14	9.9	7.1	4.4	1	90
1617	153	14	10.1	5.7	5	0.9	90



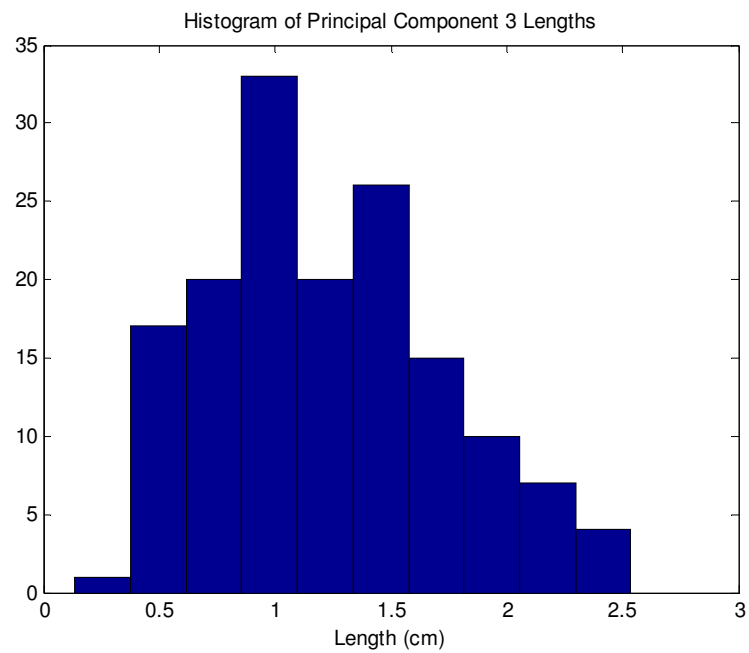
Appendix Figure 63 Histogram of Historical Plan Database Volumes



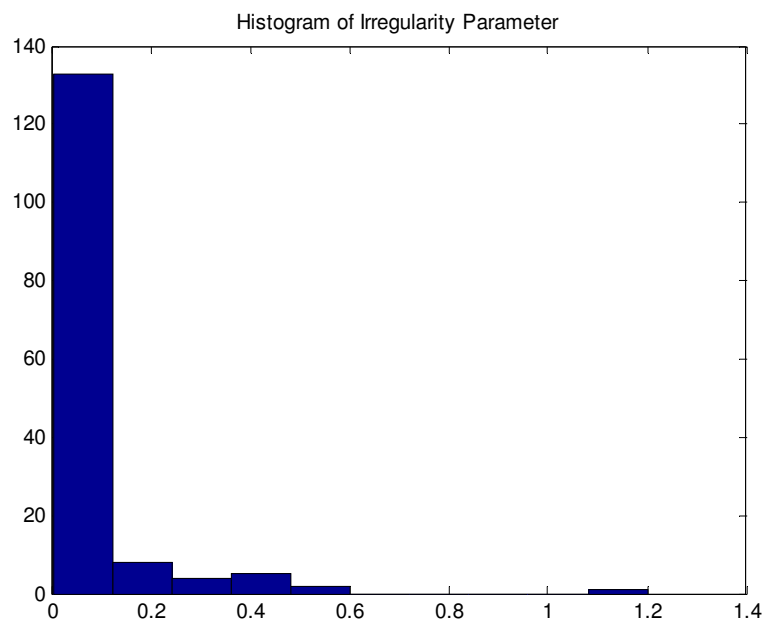
Appendix Figure 64 Histogram of Historical Plan Principal Component 1 Length



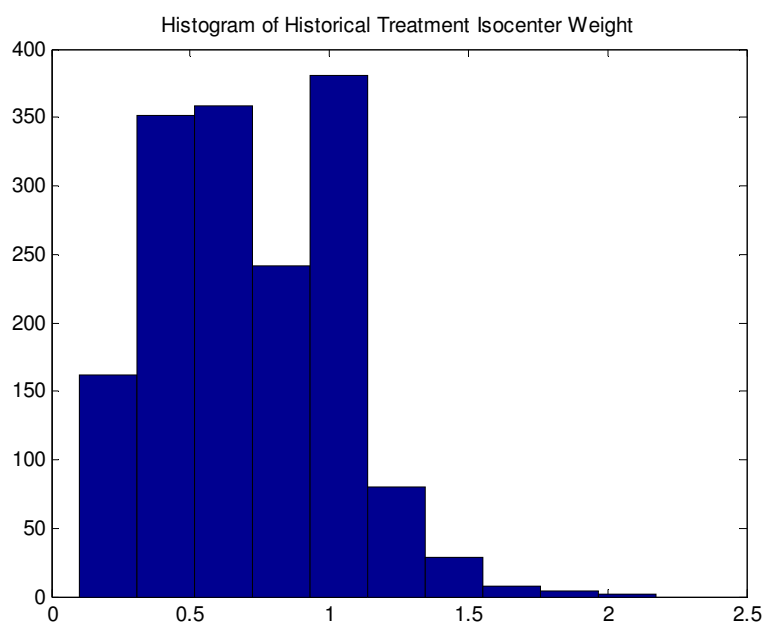
Appendix Figure 65 Histogram of Historical Plan Principal Component 2 Length



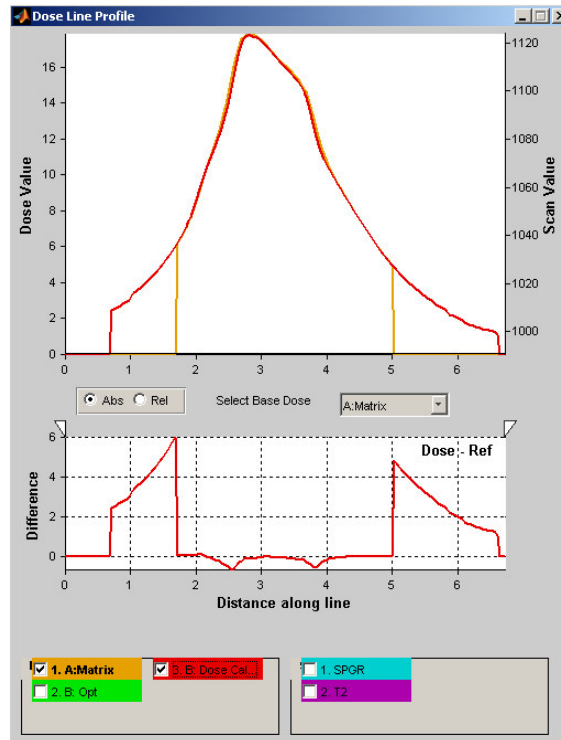
Appendix Figure 66 Histogram of Historical Plan Principal Component 3 Length



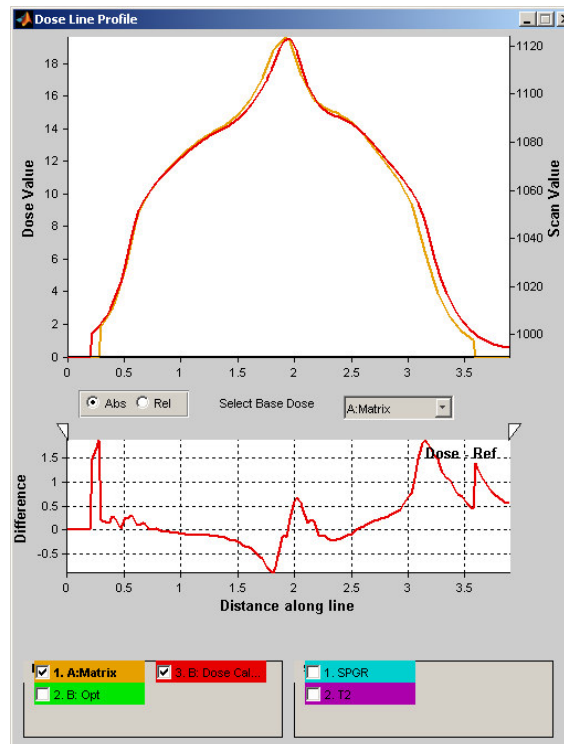
Appendix Figure 67 Histogram of Irregularity Parameter



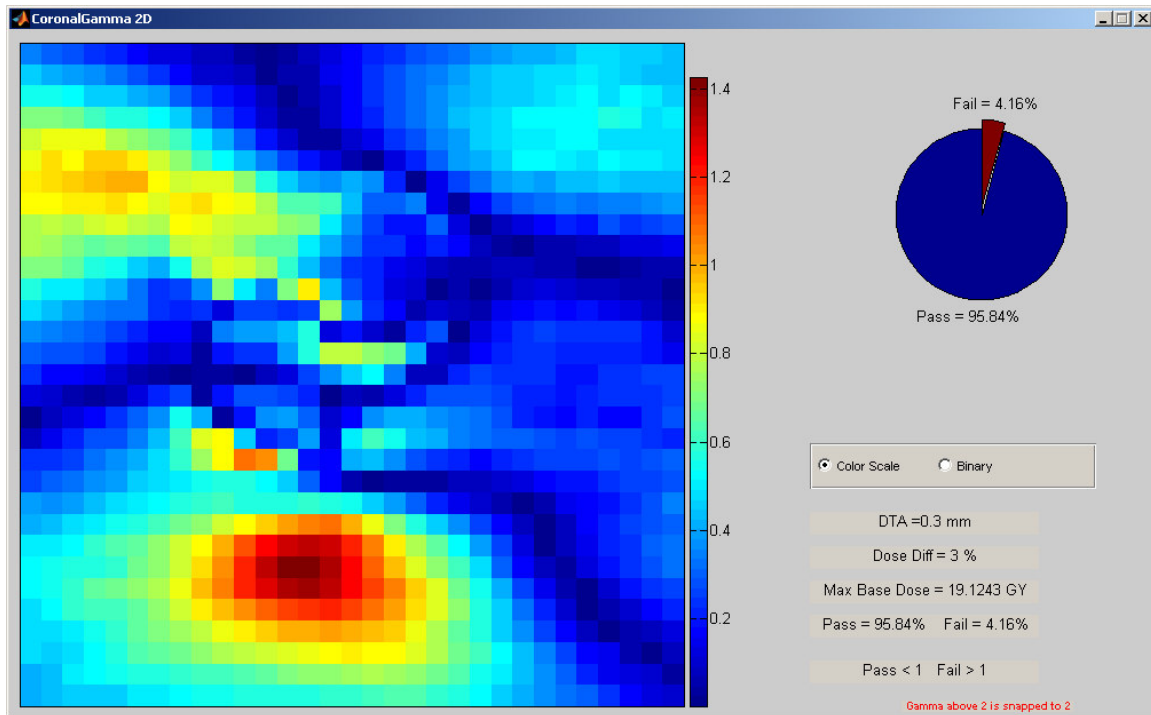
Appendix Figure 68 Histogram of Historical Plan Treatment Isocenter Weight



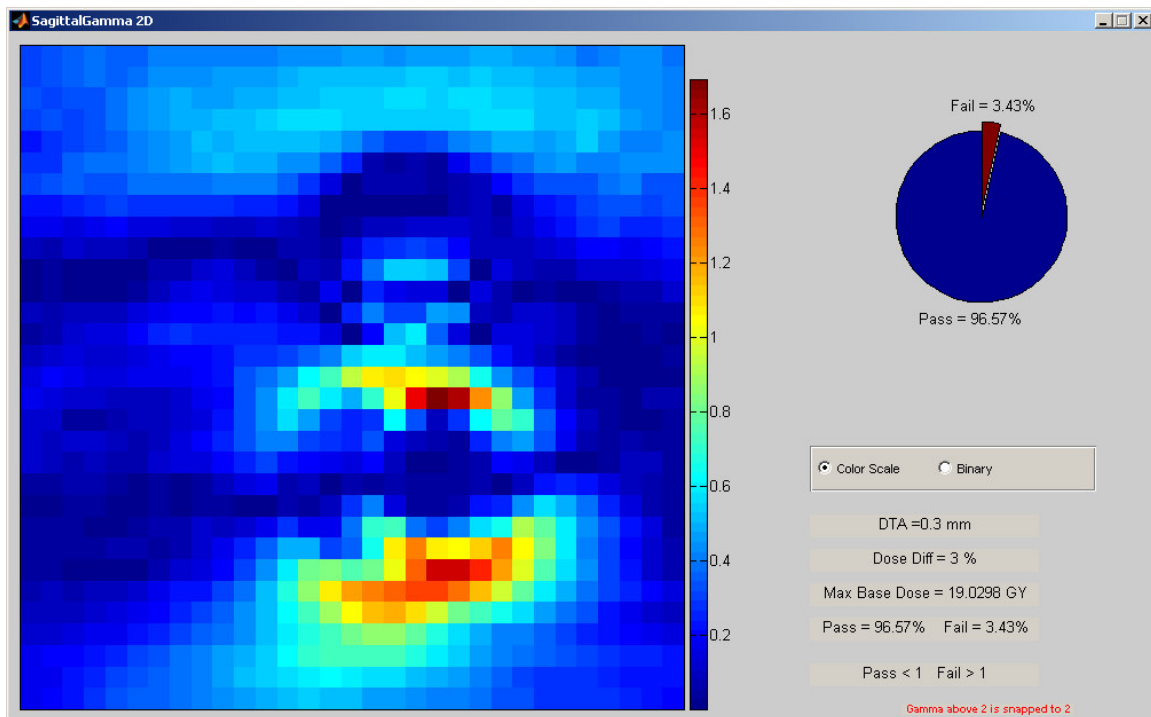
Appendix Figure 69 Gamma Plan vs. Rapid Dose Calc Comparison Y Plane



Appendix Figure 70 Gamma Plan vs. Rapid Dose Calc Comparison Z Plane



Appendix Figure 71 Gamma Plan vs Rapid Radiation Gamma Coronal Plane



Appendix Figure 72 Gamma Plan vs Rapid Radiation Gamma Sagittal Plane

Appendix C: Institutional Review Board Exemption Documentation

FORM A

Certification for Exemption from IRB Review for Research Involving Human Subjects

A. PRINCIPAL INVESTIGATOR(s) and/or CO-PI(s) (For student projects, list both the student and the advisor.):

Joseph Bowling (student), J Wes Hines (advisor)

B. DEPARTMENT: Nuclear Engineering

C. COMPLETE MAILING ADDRESS AND PHONE NUMBER OF PI(s) and CO-PI(s):

Joseph Bowling (865-769-3088)
10321 Red Water Ln
Knoxville, TN 37932

J Wes Hines (865-974-6561)
Department of Nuclear Engineering
315 Pasqua Engineering Building
Knoxville, TN 37996-2300

D. TITLE OF PROJECT: Leksell Gamma Knife Treatment Planning via Kernel Regression Data Mining Initialization and Genetic Algorithm Optimization

E. EXTERNAL FUNDING AGENCY AND ID NUMBER (if applicable): No external funding

F. GRANT SUBMISSION DEADLINE (if applicable): No grant submission

G. STARTING DATE (NO RESEARCH MAY BE INITIATED UNTIL CERTIFICATION IS GRANTED.): Upon receipt of certification

H. ESTIMATED COMPLETION DATE (Include all aspects of research and final write-up.): Summer 2010

I. RESEARCH PROJECT

1. **Objective(s) of Project** (Use additional page, if needed.): See attached page

2. **Subjects** (Use additional page, if needed.): See attached page

3. **Methods or Procedures** (Use additional page, if needed.): See attached page

4. **CATEGORY(s) FOR EXEMPT RESEARCH PER 45 CFR 46** (See instructions for categories.): IRB Exemption Category 4.

J. CERTIFICATION: The research described herein is in compliance with 45 CFR 46.101(b) and presents subjects with no more than minimal risk as defined by applicable regulations.

Principal Investigator: J. Wesley Hines J. Wesley Hines 11/25/09
Name Signature Date

Student Advisor: Joseph Bowling Joseph Bowling 11/28/09
Name Signature Date

Department Review Committee Chair: _____
Name Signature Date

APPROVED:

Department Head: _____
Name Signature Date

COPY OF THIS COMPLETED FORM MUST BE SENT TO COMPLIANCE OFFICE IMMEDIATELY UPON COMPLETION.

Rev. 01/2005

INSTRUCTIONS FOR COMPLETING FORM A
PLEASE TYPE THE INFORMATION REQUESTED ON THE FRONT OF THIS FORM

Provide the required information in the space available if at all possible. If additional space is necessary, attach a separate sheet. Submit one copy of this form to the Chair of your Departmental Review Committee for review and approval. [PLEASE NOTE: This form may be reproduced on a personal computer and printed on a high quality printer (e.g., LaserJet, DeskJet). Form A was originally created under WordPerfect 6.1 and printed on a HP LaserJet III printer using a 9-point CG Times font.]

ALL SIGNATURES MUST BE ORIGINAL on this form. When certified by your department or unit head, a copy of the signed Form A will be returned to the Principal Investigator and a copy will be returned to the Research Compliance Services Section, Office of Research.

I.1. OBJECTIVES: Briefly state, in non-technical language, the purpose of the research, with special reference to human subjects involved.

I.2. SUBJECTS: Briefly describe the subjects by number to be used, criteria of selection or exclusion, the population from which they will be selected, duration of involvement, and any special characteristics necessary to the research.

I.3. METHODS OR PROCEDURES: Briefly enumerate, in non-technical language, the research methods which directly involve use of human subjects. List any potential risks, or lack of such, to subjects and any protection measures. Explain how anonymity of names and confidentiality of materials with names and/or data will be obtained and maintained. List the names of individuals who will have access to names and/or data.

I.4. CATEGORY(S) FOR EXEMPT RESEARCH PER 45 CFR 46: Referring to the extracts below from Federal regulations, cite the paragraph(s) which you deem entitle this research project to certification as exempt from review by the Institutional Review Board. **45 CFR 46.101(b): Research activities in which the only involvement of human subjects will be in one or more of the following categories are exempt from IRB review:**

(1) Research conducted in established or commonly accepted educational settings, involving normal educational practices, such as: (i) research on regular and special education instructional strategies, or (ii) research on the effectiveness of or the comparison among instructional techniques, curricula, or classroom management methods.

(2) Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, **unless:** (i) information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; **and** (ii) any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

PLEASE NOTE: *An exemption cannot be used when children are involved for research involving survey or interview procedures or observations of public behavior, except for research involving observation of public behavior when the investigator(s) do not participate in the activities being observed. [45 CFR 46.101(b)]*

(3) Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior that is not exempt under paragraph (2) above, if: (i) the human subjects are elected or appointed public officials or candidates for public office; or (ii) Federal statute(s) require(s) without exception that the confidentiality of the personally identifiable information will be maintained throughout the research and thereafter.

(4) Research involving the collection or study of existing data, documents, records, pathological specimens or diagnostic specimens, if these sources are publicly available or if the information is recorded by the investigator in such a manner that subjects cannot be identified, directly or through identifiers linked to the subjects.

(5) Research and demonstration projects which are conducted by or subject to the approval of Federal Department or Agency heads, and which are designed to study, evaluate, or otherwise examine: (i) public benefit or service programs; (ii) procedures for obtaining benefits or services under those programs; (iii) possible changes in or alternatives to those programs or procedures; or (iv) possible changes in methods or levels of payment for benefits or services under those programs.

(6) Taste and food quality evaluation and consumer acceptance studies, if wholesome foods without additives are consumed or if a food is consumed that contains a food ingredient at or below the level and for a use found to be safe, or agricultural chemical or environmental contaminants at or below the level found to be safe, by the Food and Drug Administration or approved by the Environmental Protection Agency or the Food Safety and Inspection Service of the US Department of Agriculture.

For additional information on Form A, contact the Office of Research Compliance Officer by e-mail or by phone at (865) 974-3466.

Rev. 01/2005

I. RESEARCH PROJECT

1. Objective(s) of Project:

The objective to the research is to develop a fully automated system for performing Gamma Knife radiosurgery treatment planning. Gamma Knife is a medical device used to treat intracranial disease with ionizing radiation located at Baptist Riverside a department of Baptist Hospital West in Knoxville TN. The proposed work will in no way alter the regular treatment of the patient at the Baptist Regional Gamma Knife center and all patients are coming to the department strictly to receive a Gamma Knife treatment, not to participate in the research.

During Gamma Knife treatment a radiation plan is developed to treat the patient's disease. This plan consists of an imaging study (such as an MRI), regions of interest outlined on the imaging study (such as the desired treatment area), and a radiation treatment plan that consists of a series of coordinates and treatment times to program into the Gamma Knife device to treat the patient. The proposed work is going to explore the feasibility of producing new treatment plans by utilizing similar past treatment plans as a starting point and a Genetic Algorithm optimization module to tailor the old plans to the new plans.

Human subjects will have no direct participation in the research. All data would be acquire regardless of the propose research. A letter from the Baptist Regional Gamma Knife granting permission to use the data is attached as part of this application. The research will only make use of past treatment plans including imaging studies, regions of interest, and treatment plans. All data utilized in the research will have no patient identifiers and no way of linking the data to any patient. Also, the proposed work will in no way be use to plan or evaluate the quality of any patient treated at the Baptist Regional Gamma Knife.

2. Subjects (Use additional page, if needed.):

Research will be performed on Gamma Knife treatment planning records. All data used in the research would be acquired as part of the patient's normal Gamma Knife treatment. The Gamma Knife treatment planning records used will have no patient identifiers and there will be no way to link any patient to the records. A letter granting access to the patient records from the Baptist Regional Gamma Knife for the purpose of research is attached.

Baptist Regional Gamma Knife treats patients with intracranial disease such as brain metastasis, primary brain tumors such as glial tumors, acoustic neuromas, and pituitary adenomas, functional disorders such as trigeminal neuralgia, and vascular malformations. No pediatric patients (patients less than 18 years of age) are treated at the Baptist Regional Gamma Knife. The Gamma Knife treatment plans used in the research will include the imaging study performed as part of the Gamma Knife treatment, the mathematical representation of the regions of interested on the imaging study defining the

diseased area, and the developed Gamma Knife plan. This data will be exported from a Gamma Knife treatment planning system. The exported data will not contain any patient identifiers.

3. Methods or Procedures (Use additional page, if needed.):

The proposed work will automate the Gamma Knife treatment planning process. Gamma Knife treatment planning requires manually specifying a series of radiation beam locations and treatment times to completely irradiate a diseased area in the patient's cranium. The proposed work will automate this process by searching a dataset of past plans for several treatment volumes that are mathematically similar to the treatment volume to be planned. The radiation treatment plans that correspond to the past similar volumes are then applied to the current volume to be planned. An optimization routine known as a Genetic Algorithm is then used to tailor the past treatment plans to the current volume to produce a new acceptable treatment plan.

There are no potential risk to human subjects as the proposed system will in no way be used to clinical evaluate or produce Gamma Knife treatment plans for actual patients. The research will solely be performed on past Gamma Knife treatment planning records. All portions of these recorded utilized in research are collected as part of a normal Gamma Knife treatment.

All subjects will remain anonymous during the research. The Gamma Knife treatment records utilized in the research will not contain any patient identifiers and no way to link the records to the identity of any patient. The computer where the records are stored will be password protected. The only individual with access to the computer is the student principal investigator (Mr. Joseph Bowling).

Informed consent will not be obtained. This is because the only record linking the subject and the research would be the consent document and the principal risk would be potential harm resulting from a breach of confidentiality.

November 24, 2009

Brenda Lawson
Compliance Officer and IRB Administrator
UT Knoxville Office of Research
1534 White Ave.
Knoxville, TN 37996-1529

Dear Ms. Lawson:

This letter is to grant Mr. Joseph Bowling permission and access to Gamma Knife treatment planning records for the purpose of his dissertation research entitled "Leksell Gamma Knife Treatment Planning via Kernel Regression Data Mining Initialization and Genetic Algorithm Optimization." Please feel free to contact me with any question regarding this matter (865-549-4000).

Sincerely,



Rita Latour, CMPE
Gamma Knife Coordinator
Baptist Regional Gamma Knife Center

Joseph Bowling <bowling.joseph@gmail.com>

 Form A

Lawson, Brenda S <blawson@utk.edu>
To: "Bowling, Joseph Michael" <jbowling@utk.edu>
Cc: "Hines, J Wesley (Wes)" <jhines2@utk.edu>

Tue, Dec 1, 2009 at 1:58 PM

I have reviewed your proposed Form A human subjects' research protocol titled "Leksell Gamma Knife Treatment Planning via Kernel Regression Data Mining Initialization and Genetic Algorithm Optimization", and will certify it to be exempt from IRB review. You may proceed with your research.

Brenda Lawson
Compliance Officer and IRB Administrator
Office of Research
The University of Tennessee, Knoxville
PH: 865-974-7697
FAX: 865-974-7400

Vita

Joseph Michael Bowling was born in Lexington, Kentucky on December 8th, 1979 to his parents Michael and Belinda Bowling. He has two younger brothers Justin and Brian and a younger sister Bethany. He grew up in Richmond Kentucky and graduated high school from Madison Center High School in 1998. He entered college in August 1998 at Eastern Kentucky University where he studied Physic. While at Eastern, Joseph met his future wife Sara whom he would later marry in July of 2006. He finished his coursework at Eastern in May 2002 receiving a Bachelor of Science in Physics in May 2002. After graduating, he enrolled in graduate school at the University of Tennessee Knoxville (UTK). He received a Master of Science degree in Nuclear Engineering with a thesis entitled “Fault Detection and Isolation Expert System and Kernel Smoothing Techniques to Monitor the Continuous Automated Vault Inventory System (CAVIS)” under the guidance of J Wesley Hines, Ph.D. in May 2004. After graduation, he was began a Medical Physics clinical residency at Baptist Hospital of East Tennessee in Knoxville. Joseph continued his education at UTK by enrolling in evening classes in addition to his clinical training at the hospital. He completed his Medical Radiological Physics boards becoming a diplomat of the American Board of Radiology in May 2007. He changed employers to the Thompson Cancer Survival Center in January 2010 and oversaw the transfer of the Gamma Knife program from Baptist Hospital of East Tennessee to his new health system. He will receive a Doctor of Philosophy degree in Nuclear Engineering from the University of Tennessee in 2011 with a dissertation entitled “Leksell Gamma Knife Treatment Planning via Kernel Regression Data Mining Initialization and Genetic Algorithm Optimization.” After completing his doctoral degree, Dr. Bowling plans to

continue his current clinical work at the Fort Sanders Regional Gamma Knife center and the Thompson Cancer Survival Center.