To the Graduate Council:

I am submitting herewith a thesis written by Geoffrey C. Yerem entitled "A Computer Vision System for the Automatic Inspection of Geometric Distortions in Television Displays." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Daniel B. Koch
Major Professor

We have read this thesis
and recommend its acceptance:

Phillip W. Smith

Michael J. Roberts

Accepted for the Council:

Dr. Anne Mayhew
Vice Provost and Dean of Graduate Studies

(Original signatures are on file in the Graduate Student Services Office.)

A Computer Vision System for the Automatic Inspection of Geometric Distortions in Television Displays

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Geoffrey C. Yerem
December 2001

Acknowledgments

I would like to thank my instructors and advisors, Dan Koch, Phil Smith, Ross Whitaker, Mike Roberts, and Hamed Sari-Saraff of the University of Tennessee, Knoxville. Their instruction and encouragement had a beneficial effect on my education. I will carry on what I have learned from them for the rest of my career as an electrical engineer.

I would also like to thank Lee Hoke, Harold Hicks, Vic Degutis, Don McConnell, Terry DeWick and Zaher Samman of Philips Consumer Electronics for their support of this project. Our cooperative work on the Advanced Testing Facility was mutually beneficial.

My parents Ara and Marie Yerem deserve much recognition for encouraging my education over the years, especially during my time at university. Their love and faith in me brought me to this point.

Most importantly, I would like to offer praise to the living God, the God of Abraham, Isaac, and Jacob. I am grateful for his blessings on my family and me.

Abstract

The ability to automatically measure the image quality of a television display is a valuable resource. In display manufacturing, automatic inspection enables automatic television alignment, which reduces manufacturing costs and improves product quality. Automatic inspection also comes in handy during competitive analysis and engineering review. Primarily though, commercial inspection systems are built and used for manufacturing.

In the past two decades, the advent of microcomputers has made automatic inspection feasible through the use of computer vision. Most of the approaches developed can be divided into two groups: *fixtured systems*, with fixed or movable cameras, and *position independent systems*, which can use one or more cameras. A fixtured system involves placing the television-under-test in a rig with attached cameras. The cameras are either fixed or moved robotically. On the other hand, a position independent system allows one or more cameras to be placed anywhere as long as the cameras can view the television's image.

This thesis describes the development of a position-independent, two-camera television inspection system. Chapter One defines the problem and gives an overview of existing systems. Chapter Two covers camera calibration, the mathematical modeling of the way a camera forms an image. Camera calibration makes it possible to use off-the-shelf cameras as measurement sensors. Chapter Three discusses how to take stereo measurements from a pair of cameras. Stereo measurements taken from two or more cameras result in the determination of three-dimensional positions. Finally, Chapter Four mentions some results taken with the developed inspection system.

Table of Contents

Chapter 1 Introduction

1.1 Motivation 1
1.2 Problem Definition 2
1.3 Prior Art 4
1.3.1 Fixtured Systems 6
1.3.2 Digitally Controlled Displays 7
1.3.3 Position Independent Inspection 8
1.4 Common Approaches 10
1.5 Fundamental Problems 11
1.6 Implementation 12

Chapter 2 Camera Calibration

2.1 Introduction 14
2.2 Camera Model 14
2.3 Inverse Camera Model 17
2.4 Linear Camera Calibration 17
2.5 Nonlinear Camera Calibration 20
2.6 The Correspondence Problem 21
2.7 Implementation 27

Chapter 3 Stereo Measurements

3.1 Introduction 28
3.2 Finding the Line-of-Sight 28
3.3 Triangulation 29
3.4 Finding the Pose of the Television 30
3.5 Finding the Features of the Crosshatch Image 32
3.6 Implementation 37

Chapter 4 Results and Conclusions

4.1 Experimental Results 39
4.2 Future Work 44
4.2.1 Off-Line Camera Calibration 44
4.2.2 Correction for Refraction 44
4.2.3 Faster Execution 45
4.3 Conclusions 46

Bibliography 48

Appendix 52

Vita 66

List of Figures

Figure 1.1 Photograph of the ATF equipment room. 1

Figure 1.2 How to measure trapezoidal distortion. 2

Figure 1.3 How to measure pincushion/barrel distortion. 3

Figure 1.4 How to measure non-linearity. 4

Figure 1.5 Recommended cross-hatch pattern for 4:3 television displays. 5

Figure 1.6 Block diagram of the geometry measurement system. 13

Figure 2.1 Radial distortion: (a) barrel distortion (positive *k*); (b) pincushion distortion (negative *k*). 16

Figure 2.2 Image processing for the automatic correspondence procedure: (a) original image; (b) thresholded image; (c) removal of edge-touching blobs; (d) removal of unlikely blobs. 22

Figure 2.3 Connecting a feature (0) and its two closest non-colinear neighbors (1 and 2) with their neighbors in order to form three minimal area quadrilaterals. 23

Figure 2.4 Pattern recognition for the automatic correspondence procedure: (a) connection of minimal area quadrilaterals; (b) blobs grouped into likely facets; (c) discovery of model coordinate axes; (d) correspondence matching. 24

Figure 2.5 Connecting a feature (0) and two colinear neighbors (1 and 2) in order to group the blobs common to a target facet: (a) and (b) fail the test, while (c) and (d) pass the test. 25

Figure 2.6 Calibration step for the automatic correspondence procedure: (a) camera model from Direct Linear Transform; (b) camera model from nonlinear search. 26

Figure 2.7 Plot of radial distortion for the example calibration. 27

Figure 2.8 Screen capture of the camera calibration program. 27

Figure 3.1 Unaltered video image of a flat white field. 30

Figure 3.2 Finding a corner of the flat white field: (a) edge image; (b) initial guess; (c) optimized estimate. 32

Figure 3.3 Unaltered video image of a crosshatch. 33

Figure 3.4 Image processing for the crosshatch measurement procedure: (a) thresholded image; (b) skeleton of thresholded image; (c) mask image with labeled features; (d) edge image. 34

Figure 3.5 Sorting the grid regions: (a) initial labels and walk; (b) sorted labels. 35

Figure 3.6 Finding an initial guess for a corner of a crosshatch box: (a) polar quadrant search; (b) plot of distance from center to edge versus angle; (c) resulting guess. 35

Figure 3.7 Search for edge corners: (a) edge image; (b) initial guess; (c) optimized fit. 36

Figure 3.8 Final two-dimensional measurement of a grid intersection: the circles are the measured corners and the cross is the average of the four corners. 37

Figure 3.9 Screen captures of the stereo measurement program: (a) finding the television's pose; (b) finding the crosshatch intersections. 38

Figure 4.1 Plot of test measurements performed by the automatic geometry measurement system..... 40

Figure 4.2 Plot of deviations in measurements from their mean (the gray circle depicts a standard deviation of 0.02 inches)..... 41

Figure 4.3 Enlargement of the outline distortions in the TV image: (a) top edge; (b) left edge; (c) right edge; (d) bottom edge..... 42

Figure 4.4 Non-linearity measurements on the TV image: (a) vertical non-linearity; (b) horizontal non-linearity. 43

Figure 4.5 A simulated plot of the refraction due to glass thickness; the perceived intersection of the lines of sight falls short of the actual intersection..... 45

Chapter 1

Introduction

1.1 Motivation

From 1994 through 1997, the Communication, Information, and Signal Processing (CISP) group at the University of Tennessee, Knoxville helped to develop an Automated Testing Facility (ATF) for the local division of Philips Consumer Electronics Company (PCEC). The two groups designed the ATF to automate much of the day-to-day testing which PCEC performs on consumer television sets. Philips uses the testing to aid in engineering development as well as to compare PCEC products with similar competitive products.

The ATF, shown in Figure 1.1, consists of four racks of test equipment automatically controlled by a Sun SPARCstation 10 through an IEEE-488 General Purpose Interface Bus (GPIB). Computer software written in the LabVIEW programming environment automates the test procedures and data recording. Additionally, an X Windows graphic terminal is available for remote control of the test devices from outside the equipment room.

One common set of tests which has not been automated in the ATF is the measurement of visual geometry errors in television displays. While all of the geometry error calculations and data recording have been automated, the actual measurements must still be taken manually. This thesis project will automate

Figure 1.1 Photograph of the ATF equipment room.

the process of measuring visual geometry errors in television displays by using computer vision technology.

1.2 Problem Definition

An important aspect of the quality of any image display device is how well it reproduces an image. During the reproduction of an image by a television set, there exist many stages which introduce visible distortions. A color cathode ray tube (CRT) typically contains three electron guns, convergence purity magnet rings, a deflection yoke, a shadow mask, and a pattern of three-color phosphors affixed to a curved glass front panel. These components and others can introduce many imaging errors. It is necessary to objectively measure these errors in order to make meaningful comparisons of display quality between one television set and another.

The scope of this project is limited to the measurement of *geometric distortions*, distortions which change the shape of the intended image. Here are some common geometric distortion measurements which can be implemented as automatic tests:

Trapezoidal Distortion - This test measures how much the image resembles a trapezoid, as depicted in Figure 1.2. A percentage error is calculated using Equations (1.1).

$$TV = \frac{DC - AB}{DC + AB} \times 100\% \tag{1.1a}$$

$$TH = \frac{AD - BC}{AD + BC} \times 100\% \tag{1.1b}$$

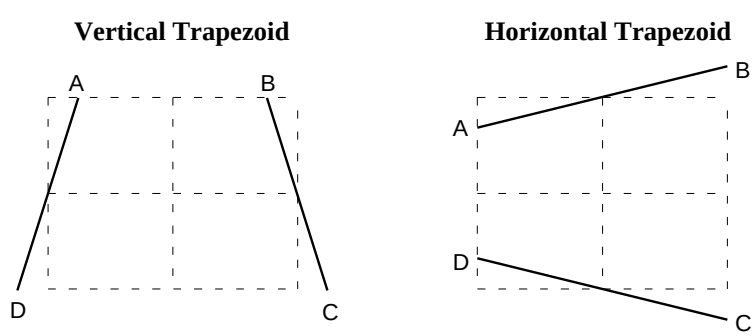


Figure 1.2 How to measure trapezoidal distortion.

Pincushion/Barrel Distortion - This test measures the amount of curvature introduced in the image produced by the television's attempt to reproduce a vertical or horizontal line. Figure 1.3 depicts pincushion distortion. The radii of curvature can be measured or a percentage error calculated using Equations (1.2).

$$CV = \frac{2 \cdot BC - FG - HJ}{2 \cdot BC} \times 100\% \tag{1.2a}$$

$$CH = \frac{2 \cdot DE - FH - GJ}{2 \cdot DE} \times 100\% \tag{1.2b}$$

Non-linearity - This test measures how uniformly a display reproduces an image across the area of its screen. The distances between adjacent horizontal/vertical lines displayed on the television are measured, as shown in Figure 1.4. The percentage differences between each distance and the mean distance are plotted using Equations (1.3).

$$DH_i = \frac{H_i - \overline{X}_h}{\overline{X}_h} \times 100\% \quad \text{where,} \quad \overline{X}_h = \frac{\sum_{i=1}^{N} H_i}{N} \tag{1.3a}$$

$$DV_j = \frac{V_j - \overline{X}_v}{\overline{X}_v} \times 100\% \quad \text{where,} \quad \overline{X}_v = \frac{\sum_{j=1}^{M} V_j}{M} \tag{1.3b}$$

All of the tests can be performed by measuring an electronically generated cross-hatch pattern reproduced on the display-under-test. The measurements must use an orthographic projection (a view from infinite distance) perpendicular to the center of the display. In other words, the IEC states that "[w]hen the

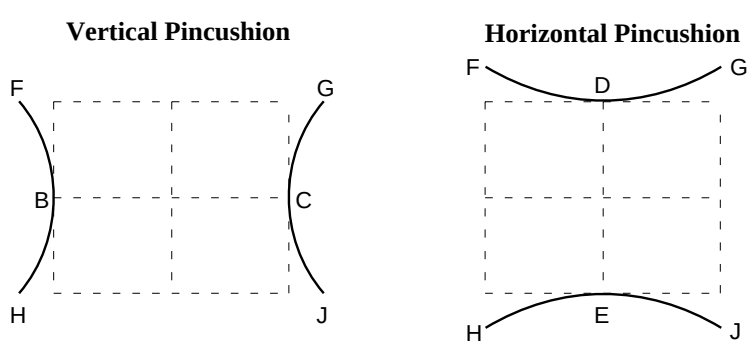


Figure 1.3 How to measure pincushion/barrel distortion.

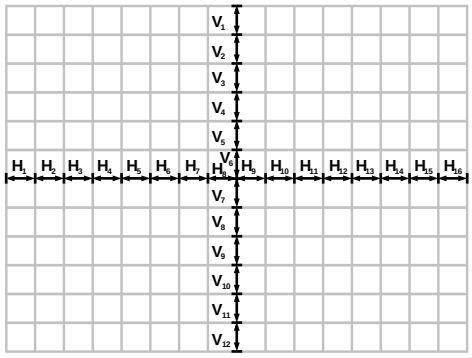


Figure 1.4 How to measure non-linearity.

screen has a curvature, like a CRT screen, the measurements shall be made on the picture projected to a virtual plane tangential to the center of the screen face." [1] The IEC specifies that the test pattern shall be a white cross-hatch on a black background, and that there shall be 13 horizontal lines and 17 vertical lines for a 4:3 aspect ratio display. Figure 1.5 illustrates a 4:3 crosshatch test pattern. This measurement technique applies to all forms of televisions including projection televisions. Liquid Crystal Display (LCD) and similar technologies do not require the measurement of geometric distortion.

The IEC also specifies that during testing, contrast, brightness, color, hue and any other controls shall be set at their normal positions or at least to the positions which lead to best picture quality. Finally, CRT displays shall be degaussed and shall face either north or south before measurements.

1.3 Prior Art

Because manufacturing drives the economics of display inspection, the most visible players in the automatic inspection field have built their devices as part of automated alignment systems for large scale assembly lines. There are many motives behind automating the inspection/alignment processes for displays. Cost: Global competition [2] is harsh and any savings in manufacturing can mean a great deal. Quality: One early application [3] used automation to accommodate the demanding specifications of military monitors. More recently [4] HDTV displays and computer monitors have demanded similar high quality adjustments. In particular, the abundance of computer monitors has increased the demand for high-

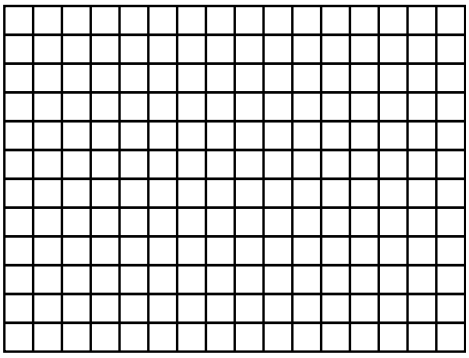


Figure 1.5 Recommended cross-hatch pattern for 4:3 television displays.

quality alignment. A typical computer monitor requires very precise alignment in order to reproduce bright, uniform, high-contrast images with straight lines and corners [2]. Speed: The alignment process poses a great bottleneck on the assembly line. Ultimately, these demanding requirements tax the ability of human operators to manually align CRT displays.

Traditionally, an operator aligned a CRT display by manually measuring the image it produced and in turn making adjustments inside. For geometry measurements, the operator typically used a template overlay [5], a tape measure [6], or a ruler with a special eyepiece for avoiding parallax, when inspecting the display. The geometry alignment of a CRT display typically occurred in two steps. In the first step, commonly known as the Integrated Tube Component (ITC) process, an operator had to adjust the color purity rings and deflection yoke on the CRT. In the second step, an operator had to adjust the display circuitry using various potentiometers on the display's circuit board. Consequently, he usually needed a mirror [5] to see the screen from behind the monitor where he made his adjustments. The adjustment process could be fairly awkward and tedious.

There are many undesirable aspects of human labor which hinders display inspection and alignment. To begin, operators require much skill [4] and experience [5] in order to perform the adjustments of displays. Besides the time and cost required to train and employ an operator to do the basic adjustments, there is a new learning curve [2] for every new design on the assembly line. Also, because the responsibility for the inspection falls upon the operator, his discretion [5] and diligence [7] affect the

resulting quality of the display. Additionally, fatigue [2, 5, 4] and stress [2] will affect the results when operators work long hours and attempt to meet deadlines and quotas. Finally, human eyesight [5, 8] cannot discriminate less than 0.33 mm [2].

Due to the nature of the human operator, manual display alignment becomes unobjective [9], inconsistent [4, 5, 7], slow [4, 7, 9], costly [8, 9], and suffers from low quality [5] and low accuracy [4].

Machines on the other hand can avoid or mitigate the failings of human operators. Artificial intelligence or optimization techniques can solve the skill and experience problems. Cost is controlled by designing an alignment system once, and then mass producing it exactly and training it quickly for new devices. Furthermore, machines have no psyche, do not become tired or scared, and can possess better "eyesight" and faster "reflexes."

1.3.1 Fixtured Systems

Initial automation [5] (ca. the 1980s) only aided a human operator by inspecting a display. Automatic inspection enforced quality and resulted in less operator movement [7]. Also, automated inspection systems could log statistics [5, 10] like failure rate, and adjustment rate for the manufacturer. This feedback improved quality assurance and provided useful documentation [11] for the manufacturer.

The earliest inspection systems used light sensors or cameras mounted on multi-axis positioners placed in close proximity to the TV. For example, in 1984 Toshiba of Japan [4] described a system using a single monochrome camera mounted on an XYZ positioner. The system could handle 9 to 26 inch televisions and gave a factor-of-ten speedup for measurements. The camera had changeable red/green/blue filters and automatic zoom and focus. By measuring three screen points, Toshiba's system could fit a sphere to the CRT and use the knowledge of the shape to automatically adjust the focus of the camera. In 1985, IBM [12] describes a similar system using a light sensor on an XY positioner.

In 1986, Mitsubishi of Japan [8] developed a system with one color camera and four monochrome cameras mounted on an XYZ positioner. Later in 1989, they used a system [13] with one monochrome camera mounted on a five-axis positioner. Because of the interactions between the adjustment controls and the huge skill required for ITC adjustment, they developed an expert system in conjunction with fuzzy set

calculations to automate the alignment. The resulting system was able to robotically perform the ITC adjustments.

In 1988, Photo Research [3] described an inspection system used for military monitor manufacturing. It used a computer with frame grabber, a camera with filter wheels, and a positioning stage. Photo Research claims to have reduced the manufacturing error of the monitor by 40%, while increasing the manufacturing speed fourfold. Their approach only used 80% of the digitized image presumably to avoid the effects of camera lens distortion. They also experimented with a cooled-integrating camera in order to sense low-light displays.

The drawbacks of a camera or light sensor on a positioner are slow speed and the need for alignment with the TV usually by a fixture [6]. This drawback is not very serious for laboratory inspection systems though.

Image Processing Systems of Canada [7, 14] patented a fixtured system built in 1994 which uses an array of cameras for automatic ITC inspection and alignment. Some of the cameras in the array focus on small areas of the screen while others focus on large areas. Also, each camera has two axes of movement to accommodate different sized displays. Partly due to an initial camera calibration step, the system can compute an orthographic projection of the television image and correct for CRT face curvature, glass thickness, and index of refraction. The system can also robotically perform ITC adjustments. Hitachi of Japan [15] in 1996 described a similar system which used multiple fixed zone cameras.

1.3.2 Digitally Controlled Displays

With the advent of digital technology, manufacturers replaced the multiple potentiometer adjustments inside CRT displays with a single-chip computer driving digital-to-analog converters. The digital chip improves on simpler analog adjustment circuits by performing more complex distortion corrections [6, 11]. Also, the digital control chip can lessen or eliminate the mechanical adjustments required in the alignment process [9]. For instance, adjustments to a digitally controlled display are made through a communication link instead of mechanical knobs. Common communication links used are RS-

232 [11], IIC bus [16], or the infrared remote control receiver [17]. Now, manufacturers commonly use digital control chips in CRT displays [18].

One early example of the digital control chip is the "Video/Timebase Processor IC" [17] presented by Motorola in 1988. It digitally controls image position, pincushion, and raster size for a CRT display. The chip stores factory settings and CRT geometry corrections.

Digital control chips can add many more controls [11] than previously used in analog alignment circuits. This is both an advantage and a problem. While, additional controls allow for finer adjustment of the display and for more complex specifications, the extra adjustments require a corresponding complexity during alignment [11]. It turns out that manual adjustment becomes very difficult if not impossible to do with a digitally controlled display. So while a digitally controlled display simplifies adjustment by an automatic alignment system, automatic alignment in turn becomes the only feasible way to adjust a digital monitor.

Around 1993, EeRise of Taiwan [9] invented the EeRise-9000, an automated alignment system centered on a digitally controlled monitor. Their goal was to make automatic alignment more workable. Their patented system [19] used one camera attached perpendicularly to a fixture composed of a dark box which an operator would manually attach to the display-under-test. Their fixture sped up the positioning process and minimized the effects of conveyor vibration.

Also, Philips [16] described a system in 1994 which used 5 cameras to align the geometry and convergence of a digitally controlled projection TV.

The digital control chip along with an automatic alignment system makes possible high quality image display without manual labor. With the digital chip and automatic alignment, manufacturers can find a balance between complexity in manufacturing and complexity in display design [10].

1.3.3 Position Independent Inspection

When an inspection system requires a fixture there are drawbacks. A robotic or human positioner is clumsy [2] and expensive. Fixtures are dependent on display size and shape, and can damage the

cosmetically sensitive areas of the television [18]. Also, shock and vibration from the assembly line can ruin the camera [18] or other sensitive parts of the inspection system.

The system that Motorola described in 1988 [17] used a single video camera with no physical contact with the display-under-test. The system had better resolution than the human eye, and was not specific to screen size. Unfortunately, they did not give many details as to how the system worked.

Display Laboratories have several patents [11, 20, 21] on a position independent, single camera inspection system [10] which uses 3D model-based computer vision. Their system can handle monochrome and color monitors from 12" to 23". The system uses a-priori knowledge of the shape of the enclosure as well as the shape of the CRT face, its thickness and index of refraction. They synchronized the camera to the test pattern generator with the camera "shutter" set for half of a TV field. During a test, the system captures two fields and then combines them into one frame [11]. The high speed capture can reduce the effects of conveyor belt vibration on the measurement [18]. Using the shadow mask, aperture grill, or display bezel, the system can find the relative pose (position and angle) of the display with respect to the camera [6]. With the pose of the display known, simple or no fixturing can be used as long as the display is fully in the field of view of the camera [6]. The system then performs a transformation on the image to form an orthographic projection which eliminates effects of perspective and parallax [6].

Around 1995, EeRise patented [19] the EeRise-9300, a contactless "compound eye system." It is a three-camera factory-calibrated stereo system mounted on a rigid base. The cameras use auto irises to adapt to the brightness and contrast of monitor. Additionally, the system has an ambient light hood with indirect lighting to avoid reflections of the light source on the screen. Their system uses one test pattern to locate the monitor and find the shape of the CRT, while using another pattern for alignment [19]. Since the system uses a general stereo approach it maintains position independence. Also, because it does not use model-based 3D, it can handle multiple monitor sizes, bezel types, and alignment specifications on the same assembly line or test bench [22]. EeRise claims that their approach surpasses model-based 3D with reduced errors in the pose calculations. They also recognize the refractive errors caused by the CRT, but do not discuss their solution to the problem [19].

Around the same time (ca. 1994), Photo Research patented a stereo inspection technique [23]. It is a fixtureless and position independent system which works on any size monitor [2]. The system takes into account monitor position, glass curvature and thickness. They transform the measurement with an orthographic projection. Because geometry adjustment controls interact, they developed a technique for measuring a linear approximation of the image changes due to each control [24]. With this information, the computer can simultaneously adjust the controls [2, 24] and find the best alignment in a few tries.

Due to modern improvements (e.g., innovative design and reduced production costs), CRTs have stayed the dominant display technology [2]. By using automated inspection/alignment systems, monitors reach the market faster because of a fast learning curve and lower defects [2]. Overall, automated alignment systems are consistent, objective, minimize assembly and test labor, increase production yield [17] and reduce the cost of manufacturing.

1.4 Common Approaches

The previous section chronicled many approaches currently used for automatic display inspection. All of the approaches apply to the cases during manufacturing (alignment) and afterward (inspection). The systems typically use one or more electronic cameras, a computer with an interface to the camera (e.g., a frame grabber), a test pattern generator (usually with a synchronizing signal to the camera or frame grabber), and a communication link to the pattern generator and digital monitor. Some of the systems use custom fixturing or robotics. Additionally, the systems can perform auxiliary tasks like commanding a programmable logic controller for an assembly line conveyor belt [14]. Altogether, two approaches emerge and can be categorized as either Fixtured or Freely Positioned.

<u>Fixtured</u> - This approach involves placing the display-under-test in a special rig fitted with the camera(s). Within this category, the cameras can be fixed or articulated. The fixed camera approach uses one or more cameras rigidly attached to the fixture with the cameras focused at different parts of the screen. On the other hand, the articulated approach typically uses a single camera fitted to a robot arm to move the camera to different parts of the screen. The robot arm usually has two or three orthogonal axes of freedom, and usually resembles a computer pen plotter.

Freely Positioned - This approach involves placing one or more cameras roughly in front of the television and uses mathematical transformations to compute the end results. If the system knows the properties of the television a-priori, then it can take the measurements with one camera. A more general approach uses two or more cameras to take stereo measurements. Either case makes a three-dimensional measurement of the display's image.

The articulated camera approach can cost a lot of money and does not use off-the-shelf components (i.e., the robot arm). Also, fixtured systems are not portable due to weight and size. As a result, the display-under-test needs to be brought to the inspection system. This can be inconvenient in a laboratory (which does not have a conveyor belt). Additionally, a fixtured system can impose a limitation on the size and shape of the display-under-test. Freely positioned systems are somewhat more general, in that they are not specific to display size and shape and do not require extensive mechanical devices.

The drawback to these systems for use in the laboratory is that they are typically designed for manufacturing. Assembly line systems typically perform measurements on a single model of television over and over again with the intention of automatic alignment. Operating on a single model simplifies the problem somewhat, since the measurement system can be configured to anticipate all of the properties of that television. For the pure inspection problem though, such systems can be expensive, scarce, and difficult to integrate or customize. Because they are used in manufacturing though, their high cost is not significant compared to the savings in manufacturing cost. Yet, in a laboratory, geometry measurements compose a small amount of the overall tests performed and should be proportionately affordable.

1.5 Fundamental Problems

A flexible, multi-TV inspection/alignment system is difficult to implement [2]. For example, the curved glass of the CRT face, can be 20-30 mm thick [4] which causes parallax [11].

Every approach must deal with the following two problems associated with the measurement of CRT displays. First, they must calibrate the measuring instrument to determine its pose relative to the unit-under-test as well as the instrument's internal properties. Photogrammetrists call these the *external* and *internal* parameters of the instrument. Secondly, a measurement approach needs to account for the

properties of the display screen. This includes the shape of the screen and its refractive properties, like the index of refraction and the glass thickness, which can vary over the surface.

The fixtured approaches solve the pose problem by placing the unit-under-test in a special rig which fixes the orientation of the measurement instrument and the TV. A second benefit emerges by choosing a special orientation in order to make the camera perpendicular to the TV and solve the display screen problem. For the fixed camera approaches, orthographic projections can be approximated for the small regions of the screen each camera focuses on. For the articulated camera approach, the system moves the camera to the area of interest and measures the position of the camera.

The freely positioned camera systems are divided into two approaches, single camera and stereo. In the single camera approach, knowledge of the TV is assumed and the pose of the camera is determined from the knowledge of the shape of the TV. Then the measurements are taken using knowledge of the properties of the screen. For the stereo approach, knowledge of the shape of the screen is not necessary but its refractive properties are still important.

1.6 Implementation

For the system described herein, the stereo approach was chosen. The stereo approach allows the cameras to be freely and independently positioned. As a result, the cameras do not need to be mounted on a special (i.e., expensive) rig. Also, the stereo approach makes the calibration system somewhat portable in that the cameras can be brought to the display-under-test. Overall, the system was easy to build with off-the-shelf components. In fact, because most of the components were readily available, the only item that had to be purchased for inclusion with this system was a frame grabber expansion board.

The system revolves around a Sun SPARCstation 10 computer with a SunVideo frame grabber. The frame grabber has three switched inputs, of which one can be used at any time. Even though the frame grabber can only capture an image from one camera at a time, this is not a problem because the scene is static. Also, two Apple VideoPhone cameras with NTSC composite video outputs, each mounted on its own tripod, were used since they were inexpensive and readily available. Unfortunately, with low cost comes a low quality, high distortion lens. To deal with the lens distortion and independently positioned

cameras, a calibration target is used to find the pose and internal properties of the cameras. Finally, a computer with a composite video output generates the test patterns. The software for the system was implemented in ARM C++ and used the Solaris XIL Imaging Library. Figure 1.6 depicts a block diagram of the system.

The following steps are required in order to perform a geometry test on a display with this system. First, the tripod mounted cameras are positioned with overlapping views so they can both see the television screen. Then the television is moved away and a calibration target is placed in front of the cameras. A camera calibration computer program finds the pose of the cameras relative to one another as well as their internal properties. Next the calibration target is removed and the television is placed in front of the cameras. The ambient light is also turned off to eliminate reflections in the CRT screen. With a flat white field displayed on the television, a pose estimation computer program finds the pose of the television relative to the cameras. Finally, with a crosshatch pattern displayed on the television, a stereo measurement computer program performs the actual geometry measurements and inspection.

The next two chapters in this thesis will describe the details of the camera calibration and stereo measurements respectively, while the concluding chapter will describe results from an experiment using this system.

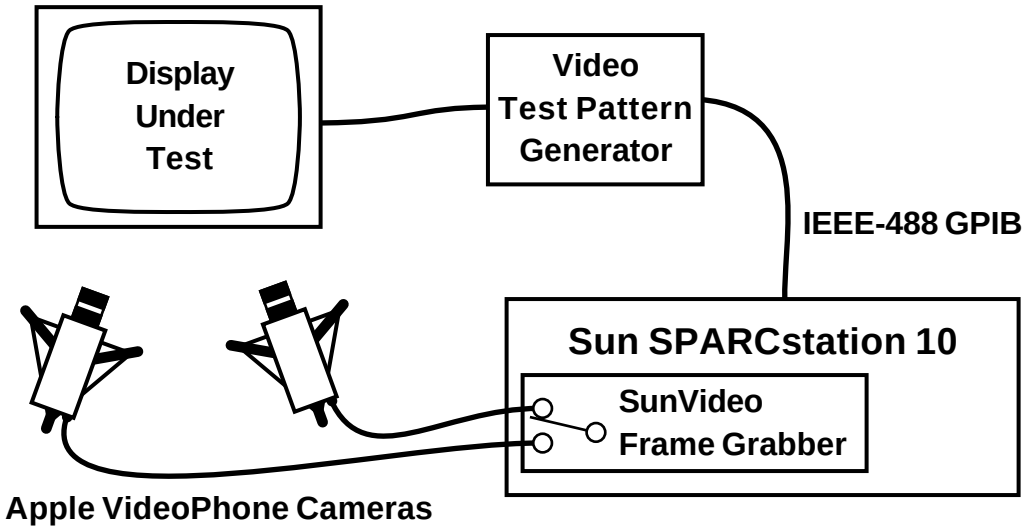


Figure 1.6 Block diagram of the geometry measurement system.

Chapter 2

Camera Calibration

2.1 Introduction

Camera Calibration is the process of finding a mathematical model which represents the way a camera forms an image. This allows one to match any location in the image with a *line-of-sight* in the real world. The line-of-sight is a ray which extends from the camera and includes every point in the world which could possibly be projected onto a point in an image. In other words, a many-to-one relationship exists between the possible 3D points along the ray and the single 2D imaged point.

For conventional cameras, photogrammetrists use a perspective projection model commonly known as the *pinhole camera model*. The pinhole model has *internal parameters*, like focal length and principal point as well as *external parameters*, such as the camera's position and rotation in the world coordinate frame. While most manufacturers design cameras to follow the idealized pinhole model, the camera's lens system and its wider-than-a-pinhole aperture typically creates discrepancies or *distortion*. As a result, the camera model additionally includes internal parameters which describe the distortion.

Camera calibration is essential in analytic photogrammetry since it enables a photogrammetrist to interpret the 3D world from 2D images. Tsai [25] points out that camera calibration also results in the ability to use off-the-shelf cameras and lenses which makes photogrammetry affordable and casual.

2.2 Camera Model

This project employs a conventional camera model [25, 26, 27]. To begin, Equation (2.1) represents the external model of the camera, or the transformation from a point $[X \;\; Y \;\; Z]^{\mathrm{T}}$ in the world frame, to a point $[P \;\; Q \;\; S]^{\mathrm{T}}$ in the camera frame.

$$\begin{bmatrix} P \\ Q \\ S \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{T} \tag{2.1}$$

In Equation (2.1), **R** is the camera rotation in the world frame while **T** is the world origin in the camera frame. This formulation simplifies some of the mathematics later on.

The rotation, **R**, is a 3-by-3 orthonormal matrix. One representation for **R** is the *Euler transform*, a sequence of three clockwise rotations; one about the X-axis of angle ω, one about the Y-axis of angle ϕ and one about the Z-axis of angle κ.

$$\mathbf{R}(\omega,\phi,\kappa) = \mathbf{R}_Z(\kappa)\mathbf{R}_Y(\phi)\mathbf{R}_Z(\omega) \tag{2.2}$$

where,

$$\mathbf{R}_X(\omega) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \tag{2.3a}$$

$$\mathbf{R}_Y(\phi) = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{2.3b}$$

$$\mathbf{R}_Z(\kappa) = \begin{bmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3c}$$

so,

$$\mathbf{R}(\omega,\phi,\kappa) = \begin{bmatrix} \cos\phi\cos\kappa & \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa & \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa \\ -\cos\phi\sin\kappa & \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa & \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa \\ \sin\phi & -\sin\omega\cos\phi & \cos\omega\cos\phi \end{bmatrix} \tag{2.4}$$

Conversely, $[\omega,\phi,\kappa]$ can be obtained from **R** using Equations (2.5).

$$\omega = \arctan2(-r_{32}, r_{33}) \tag{2.5a}$$

$$\phi = \arcsin(r_{31}) \tag{2.5b}$$

$$\kappa = \arctan2(-r_{21}, r_{11}) \tag{2.5c}$$

The next two equations depict the internal model of the camera. The first, Equation (2.6) performs the perspective projection and adjusts for lens distortion with one term of radial distortion, *k*.

$$\begin{bmatrix} P/S \\ Q/S \end{bmatrix} = \left[1 + k\left(u^2 + v^2\right)\right]\begin{bmatrix} u \\ v \end{bmatrix} \tag{2.6}$$

Radial distortion, which causes barreling or pincushioning, is the most dominant component of distortion. It can cause straight 3D lines to project to 2D curves instead of projecting to straight lines as expected by the pinhole model. Figure 2.1 shows an example of barreling and pincushioning.

This project employs a simple distortion model in order to have an operator-friendly on-site calibration procedure. In a laboratory setting, higher-order radial distortion terms as well as decentering distortion [26, 28, 29] terms could be measured to provide a better model of the camera.

The second half of the internal camera model, Equation (2.7), converts from the distorted projection $[u \;\; v]^{\mathrm{T}}$ to the actual image measurements $[x \;\; y]^{\mathrm{T}}$.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dfrac{x - x_0}{f_x} \\ \dfrac{y - y_0}{f_y} \end{bmatrix} \tag{2.7}$$

The pair $[x_0 \;\; y_0]$, called the *principal point*, is the projection of the *perspective center* along the camera's Z-axis onto the image plane. The pair $[f_x \;\; f_y]$ is a combination of the camera's focal length in pixels and the image scaling factor, or $[f \cdot s_x \;\; f]$. The scaling factor s_x accounts for the possibility of a non-square imaging surface or deviations in the sampling of horizontal video lines by a frame grabber [30].

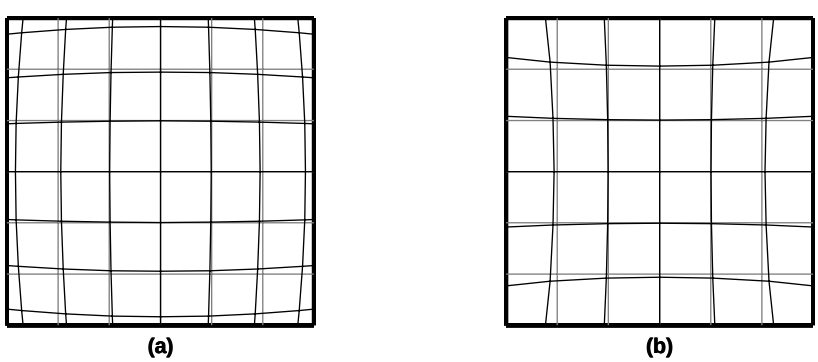


Figure 2.1 Radial distortion: (a) barrel distortion (positive *k*); (b) pincushion distortion (negative *k*).

2.3 Inverse Camera Model

Due to the nature of Equation (2.6), there is a direct path from an image point $[x \;\; y]^{\mathrm{T}}$ to the line-of-sight, which accommodates taking measurements from the image. Unfortunately, there is no simple way to take a world point $[X \;\; Y \;\; Z]^{\mathrm{T}}$ and compute an image point $[x \;\; y]^{\mathrm{T}}$ which would aid in debugging. A nonlinear numerical solution does a good job of overcoming this obstacle.

In order to obtain $[x \;\; y]^{\mathrm{T}}$ from $[X \;\; Y \;\; Z]^{\mathrm{T}}$, first project $[X \;\; Y \;\; Z]^{\mathrm{T}}$ onto the camera plane using Equation (2.1) in order to get $[P/S \quad Q/S]^{\mathrm{T}}$. Then set Equation (2.6) to zero.

$$\begin{bmatrix} \varepsilon_{P/S}(u,v) \\ \varepsilon_{Q/S}(u,v) \end{bmatrix} = \begin{bmatrix} P/S \\ Q/S \end{bmatrix} - \left[1 + k\left(u^2 + v^2\right)\right] \begin{bmatrix} u \\ v \end{bmatrix} = 0 \tag{2.8}$$

Equation (2.8) can be solved using the iterative Newton-Raphson method [31],

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} - \mathbf{J}^{-1}(u_n, v_n) \begin{bmatrix} \varepsilon_{P/S}(u_n, v_n) \\ \varepsilon_{Q/S}(u_n, v_n) \end{bmatrix} \tag{2.9}$$

where $\mathbf{J}(u,v)$ is the Jacobian of Equation (2.8).

$$\mathbf{J}(u,v) \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} = -\begin{bmatrix} 1 + k\left(3u^2 + v^2\right) & 2kuv \\ 2kuv & 1 + k\left(u^2 + 3v^2\right) \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.10}$$

Since the distorted positions are close to their original positions, $[P/S \quad Q/S]^{\mathrm{T}}$ can be used for the initial guess, $[u_0 \;\; v_0]^{\mathrm{T}}$. With single-precision (32-bit) math, the solution converges in two-to-three iterations.

Finally, Equation (2.7) converts $[u \;\; v]^{\mathrm{T}}$ into the image coordinates $[x \;\; y]^{\mathrm{T}}$.

A closed form solution happens to exist for this problem, but it is requires the evaluation of several transcendental functions which winds up being more computationally expensive. Also, this problem can be represented in a one-dimensional form, but the simplified one-dimensional form does not scale-up with more extensive distortion models, like decentering distortion.

2.4 Linear Camera Calibration

A common way to calibrate a camera is to take a picture of a 3D object which has features with known measurements. Points extracted from corners, intersections, or centroids, for example, make good

features. Consequently, calibration procedures frequently use the known 3D locations of points on a test object and the measurements of the corresponding 2D projections in a picture to find the camera's internal and external parameters.

Performing a camera calibration with point-to-point correspondences requires a mathematical relationship between the world/image pairs. To obtain one such relationship, combine Equation (2.1), (2.6), and (2.7) while setting the distortion constant *k* to zero, to form the *colinearity equations* (2.11).

$$\frac{x - x_0}{f_x} = \frac{r_{11}X + r_{12}Y + r_{13}Z + t_X}{r_{31}X + r_{32}Y + r_{33}Z + t_Z} \tag{2.11a}$$

$$\frac{y - y_0}{f_y} = \frac{r_{21}X + r_{22}Y + r_{23}Z + t_Y}{r_{31}X + r_{32}Y + r_{33}Z + t_Z} \tag{2.11b}$$

Abdel-Aziz and Karara [32] were able to reduce the colinearity equations to Equations (2.12), also known as the *Direct Linear Transform* (DLT).

$$x + \frac{L_1X + L_2Y + L_3Z + L_4}{L_9X + L_{10}Y + L_{11}Z + 1} = 0 \tag{2.12a}$$

$$y + \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1} = 0 \tag{2.12b}$$

or

$$L_1X + L_2Y + L_3Z + L_4 + L_9xX + L_{10}xY + L_{11}xZ = -x \tag{2.13a}$$

$$L_5X + L_6Y + L_7Z + L_8 + L_9yX + L_{10}yY + L_{11}yZ = -y \tag{2.13b}$$

The DLT provides a linear relationship between the world/image pairs and contains 11 unknowns. A collection of world/image pairs forms the non-homogeneous linear system,

$$\begin{bmatrix} & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & x_nX_n & x_nY_n & x_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & y_nX_n & y_nY_n & y_nZ_n \\ & & & & & \vdots & & & & & \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \\ L_6 \\ L_7 \\ L_8 \\ L_9 \\ L_{10} \\ L_{11} \end{bmatrix} = \begin{bmatrix} \vdots \\ -x_n \\ -y_n \\ \vdots \end{bmatrix} \tag{2.14}$$

which is in the form $\mathbf{AL} = \mathbf{b}$. If N is the number of measurements, then $\mathbf{A}$ is $2N \times 11$ and $\mathbf{b}$ is $2N \times 1$. Any linear least-squares solver [31] like QR Decomposition or Singular Value Decomposition can find a good solution for $\mathbf{L}$. In order to solve for the 11 unknowns, N must be at least 6, but usually a large number of points are used to create an overdetermined linear system. Moreover, it is important to note that a strictly coplanar set of target points requires a formulation [25, 33] different from Equations (2.12), since coplanar features will reduce the rank of $\mathbf{A}$ to 8.

As is, the result vector $\mathbf{L}$ from the DLT is all that is needed to find a line-of-sight. Additionally, the DLT gives a linear fit to the lens distortion. Sometimes though, it is necessary to extract the internal and external camera parameters from $\mathbf{L}$.

Common techniques for extracting the camera parameters from $\mathbf{L}$ [26, 34] revolve around using the orthonormal property of $\mathbf{R}$. Comparing Equations (2.11) with Equations (2.12) gives,

$$\vec{s}_1 = t_Z \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = -f_x \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} - x_0 \begin{bmatrix} r_{31} \\ r_{32} \\ r_{33} \end{bmatrix} \tag{2.15a}$$

$$s_2 = t_Z L_4 = -f_x t_X - x_0 t_Z \tag{2.15b}$$

$$\vec{s}_3 = t_Z \begin{bmatrix} L_5 \\ L_6 \\ L_7 \end{bmatrix} = -f_y \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} - y_0 \begin{bmatrix} r_{31} \\ r_{32} \\ r_{33} \end{bmatrix} \tag{2.15c}$$

$$\vec{s}_4 = t_Z L_8 = -f_y t_Y - y_0 t_Z \tag{2.15d}$$

$$\vec{s}_5 = t_Z \begin{bmatrix} L_9 \\ L_{10} \\ L_{11} \end{bmatrix} = \begin{bmatrix} r_{31} \\ r_{32} \\ r_{33} \end{bmatrix} \tag{2.15e}$$

To begin extracting the camera parameters, use Equation (2.16) to find the depth of the target from the camera. Because the sign of t_Z is ambiguous, it is necessary to know ahead of time whether the origin of the target is in front of or behind the camera.

$$t_Z = \frac{\pm 1}{\sqrt{L_9^2 + L_{10}^2 + L_{11}^2}} \tag{2.16}$$

Then multiply t_Z with the **L** vector. The camera parameters emerge with:

$$x_0 = -\vec{s}_1^{\mathrm{T}}\vec{s}_5 \tag{2.17a}$$
$$y_0 = -\vec{s}_3^{\mathrm{T}}\vec{s}_5 \tag{2.17b}$$
$$f_x = \left\|\vec{s}_1 + x_0\vec{s}_5\right\| \tag{2.17c}$$
$$f_y = \left\|\vec{s}_3 + y_0\vec{s}_5\right\| \tag{2.17d}$$
$$\mathbf{T} = \begin{bmatrix} -(s_2 + x_0 t_Z)/f_x & -(s_4 + y_0 t_Z)/f_y & t_Z \end{bmatrix}^{\mathrm{T}} \tag{2.17e}$$
$$\mathbf{R} = \begin{bmatrix} -(\vec{s}_1 + x_0\vec{s}_5)/f_x & -(\vec{s}_3 + y_0\vec{s}_5)/f_y & \vec{s}_5 \end{bmatrix}^{\mathrm{T}} \tag{2.17f}$$

While this technique does not guarantee an orthonormal $\mathbf{R}$, the representation is equivalent to the representation generated by the DLT.

2.5 Nonlinear Camera Calibration

To calibrate a camera with an explicit distortion model, one that keeps $\mathbf{R}$ orthonormal, requires performing an iterative nonlinear search. The approach of world/image point correspondences as in the previous section can be used to search for an optimal set of camera parameters which minimizes some error criterion. Tsai [25] categorizes this as Full-Scale Nonlinear Optimization.

The distance between the undistorted projection of a 3D point and the distortion corrected image point is a reasonable error criterion. Given the model coordinate $[X\ Y\ Z]^{\mathrm{T}}$ and its corresponding measurement $[x\ y]^{\mathrm{T}}$, use Equations (2.1) and (2.4) to get the undistorted projection $[P/S\quad Q/S]^{\mathrm{T}}$ and Equation (2.7) to get $[u\ v]^{\mathrm{T}}$. Then use Equation (2.8) to find the error. Do this for each world/image pair to find a vector $[t_X, t_Y, t_Z, \omega, \phi, \kappa, x_0, y_0, f_x, f_y, k]$ which minimizes the summed squared error as in Equation (2.18).

$$\min \sum_n \left(\varepsilon^2_{P_n/S_n} + \varepsilon^2_{Q_n/S_n}\right) \tag{2.18}$$

The linear calibration method of the previous section provides a good initial vector for this method. The procedure uses the results of the linear method by finding the Euler angles $[\omega, \phi, \kappa]$ using Equations (2.5), and by initialing the distortion, k, to zero. Also, because the optimizer is varying the Euler angles instead of directly manipulating the elements of **R**, this method results in an orthonormal **R**.

Most prepackaged multidimensional nonlinear function optimizers can solve this problem. For example, the downhill simplex method [31] does a good job. Due to the complexity and subtle tricks that nonlinear optimization requires, it is always wise to use a proven, prewritten software library.

2.6 The Correspondence Problem

Using the linear or nonlinear calibration techniques in the previous sections requires a knowledge of the correspondences between the 3D features of a calibration target and their resulting 2D projections in an image. In the worst case, human interaction can solve the problem where the user manually sorts out the correspondences. Yet, with a lot of feature points manual sorting is tedious. This project uses an automated procedure for finding the correspondences and then feeds them to the camera calibration procedures.

This technique uses a three-sided convex box as a calibration target, where each side has a 6-by-6 grid of black circles on a white background. This makes for 108 discrete features across three orthogonal facets. The circles are separated by 2 inches in each direction with a 2-inch gap at each facet edge and a 4-inch space along the outsides. This arrangement aids with the pattern recognition stage. Figure 2.2 (a) shows an unaltered video capture of the target.

The first stage in finding the 3D-to-2D correspondences involves performing a sequence of image processing steps. The first step thresholds the original image and erases any stray pixels, as shown in Figure 2.2 (b). The procedure uses a threshold of 1/2 of the image's maximum intensity value because of the purely black-and-white calibration target. Also, erasing stray pixels with no neighbors lessens the burden of the connected component analysis [35] which comes next. The connected component analysis labels the distinct groups of connected pixels or blobs in the image.

In order to ignore any clipped features at the edge of the image, the procedure erases any blobs which touch the edge as shown in Figure 2.2 (c).

The final step in the image processing stage detects and erases any unlikely blobs, which Figure 2.2 (d) depicts. Using the metrics of blob extents and blob area, certain unlikely blobs are culled from the set of remaining blobs. The extents of a blob consists of the horizontal width (w) and vertical

height (h), which encloses all of the pixels of the blob, while the area (a) comprises the number of pixels contained in the blob. The condition in Equation (2.19) contains thresholds derived from statistics collected from multiple video captures of the calibration target. The procedure uses the condition to cull the undesirable blobs.

$$\text{if } \left(a < 50 \text{ or } a > 5000 \text{ or } h < 0.1w \text{ or } h > 3w \text{ or } a < 0.4wh \text{ or } a > 1.1wh\right) \text{ then Cull} \quad (2.19)$$

The first stage finishes by computing the centroids of the selected blobs and passing the centroids to stage two. In the example portrayed in Figure 2.2, all of the fully visible blobs from the target remain, as well as a few extraneous blobs.

The second stage of the automatic camera calibration involves recognizing the target and identifying its features. Using the computed centroids of the blobs found in the image processing stage, the first step searches for the two closest centroids to each candidate blob so that the resulting three points are

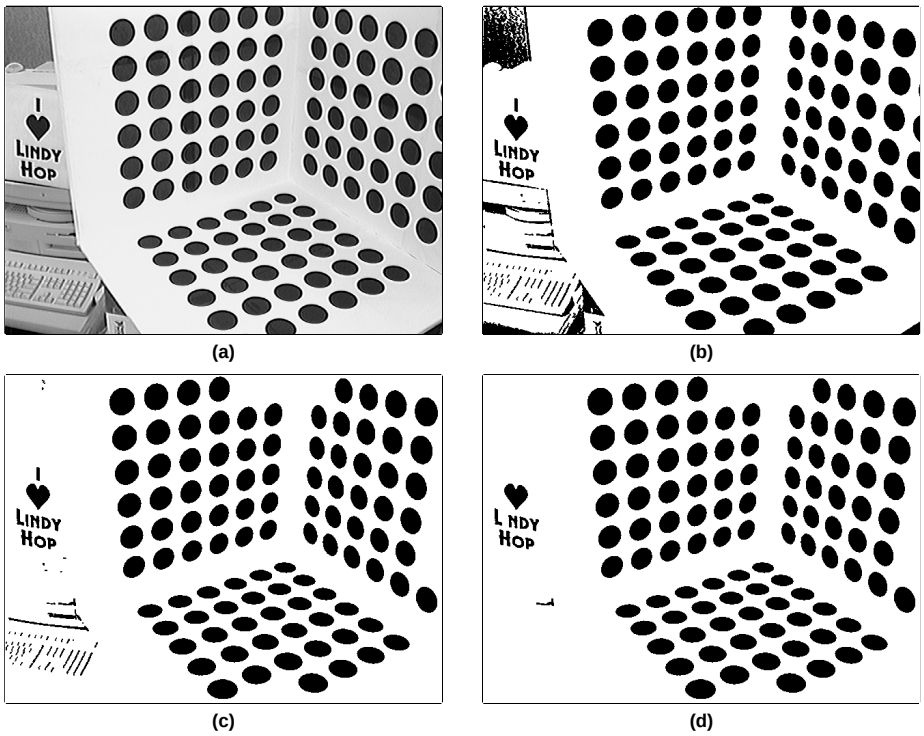


Figure 2.2 Image processing for the automatic correspondence procedure: (a) original image; (b) thresholded image; (c) removal of edge-touching blobs; (d) removal of unlikely blobs.

not colinear. Then it looks for three additional points which make up three quadrilaterals and if successful, it connects the blobs by the edges of the quadrilaterals. Figure 2.3 shows an example of this process, while Figure 2.4 (a) depicts the end result.

Next the procedure tries to group the blobs into common target facets. Blobs which share a common facet on the calibration target tend to be connected to other blobs on the same facet by line segments with similar lengths and directions, despite image distortion. Consequently, for each blob the procedure searches for two neighbors which make a common line with the original blob. If the angle between the line segments and their distances are within the range shown in Equation (2.20), the program connects the three blobs and gives them the same facet label. Figure 2.5 shows some examples of this process.

$$\text{if } \left(\frac{\min(\|d_1\|, \|d_2\|)}{\max(\|d_1\|, \|d_2\|)} > 0.86 \text{ and } \frac{d_1 \bullet d_2}{\|d_1\| \cdot \|d_2\|} < -0.999 \right) \text{ then Connect} \qquad (2.20)$$
$$\left(\text{where } d_1 = p_1 - p_0, d_2 = p_2 - p_o\right)$$

Observe in Figure 2.4 (b) that the target facets are grouped correctly and that the stray blobs are grouped alone. The procedure then keeps the three groups with the largest number of blobs.

The third step of this stage involves finding three axes which separate the target facets. This step uses the edges which connect blobs between the identified facets. A 2D line fitted through the midpoints of those edges tends to separate the facets well. Also, the intersection of the three resulting lines gives an

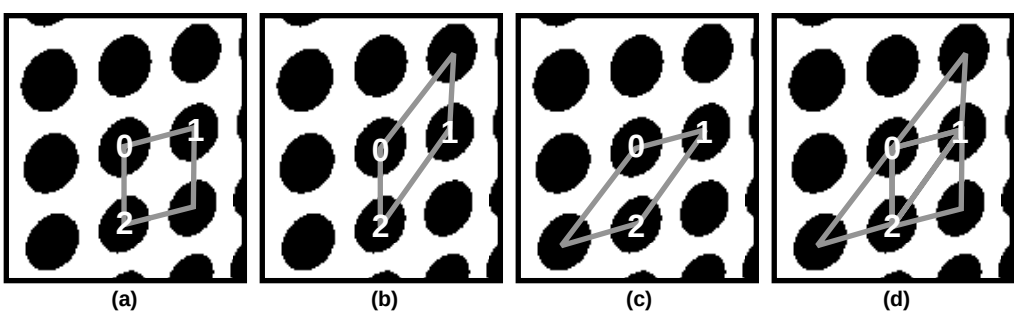


Figure 2.3 Connecting a feature (0) and its two closest non-colinear neighbors (1 and 2) with their neighbors in order to form three minimal area quadrilaterals.

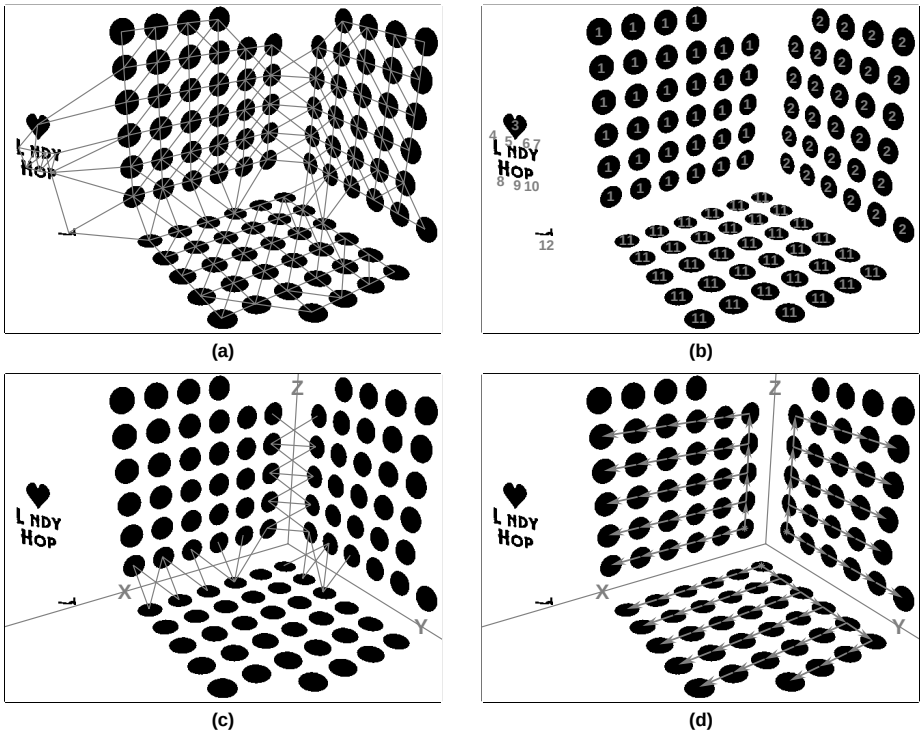


Figure 2.4 Pattern recognition for the automatic correspondence procedure: (a) connection of minimal area quadrilaterals; (b) blobs grouped into likely facets; (c) discovery of model coordinate axes; (d) correspondence matching.

approximate 2D position of the projection of the target origin. This step also sorts the axes to make a right-handed coordinate system where the Z-axis is up, the Y-axis points to the right and the X-axis is the line left-over. Figure 2.4 (c) shows the three chosen axes.

The final step in the pattern recognition stage walks the blobs and corresponds the 2D blob centroids with the 3D model of the target. For each identified facet, the walk starts with the blob closest to the located 2D target origin. The traversal proceeds in the direction of the two axes for that facet. Figure 2.4 (d) shows the results of the walks. The procedure does not try to locate every visible feature since it can usually pick up any missed correspondences in the next and final stage, the camera calibration stage.

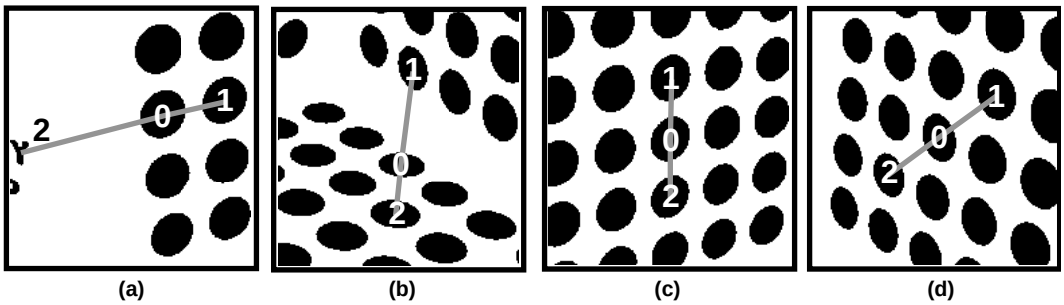


Figure 2.5 Connecting a feature (0) and two colinear neighbors (1 and 2) in order to group the blobs common to a target facet: (a) and (b) fail the test, while (c) and (d) pass the test.

The final stage of the automatic calibration procedure calculates the internal and external parameters of the camera. Given the 2D-to-3D correspondences, the solution of the DLT followed by parameter extraction provides an initial approximation to the camera parameters. Figure 2.6 (a) shows that the DLT gives a good fit for the example image. Using the linear camera model, correspondences which were missed can be picked up by using the projection of the model of the target to make a better set of 2D-to-3D correspondences. At that point, a global nonlinear search refines the linear model and finds a better fit in order to make the rotation matrix orthonormal and deal with the lens distortion. Figure 2.6 (b) depicts the final calibration results. The differences between the initial guess and the final result are small. One noticeable difference is that the lines of the projected model from the linear result are straight while the lines of the nonlinear result are slightly curved. Also, the principal point (represented by a circle) moved around the image center (represented by a cross).

The experiment portrayed in this section used an Apple VideoPhone camera fed into a Sun Video frame grabber card. Here is a list of the camera parameters obtained:

R = (119.227 -26.5886 -165.095) degrees
T = (6.33882 0.836392 39.1615) inches
o = (287.432 236.828) pixels
f = (806.104 804.422) pixels
k = 0.234107
sx = 1.00209
RMS Error = 0.584924 pixels for 98 points

The plot in Figure 2.7 shows the distortion curve of the camera.

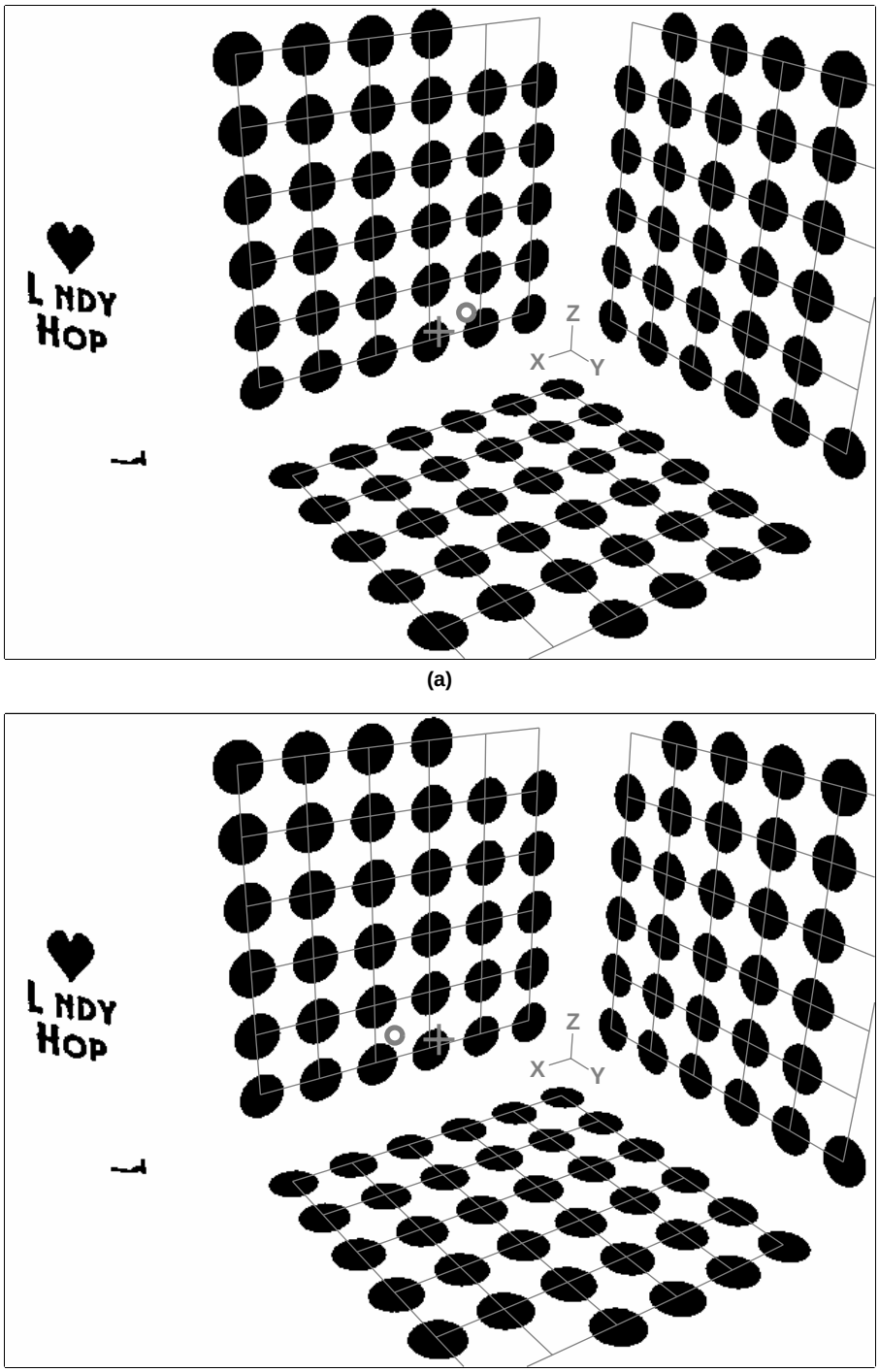


Figure 2.6 Calibration step for the automatic correspondence procedure: (a) camera model from Direct Linear Transform; (b) camera model from nonlinear search.

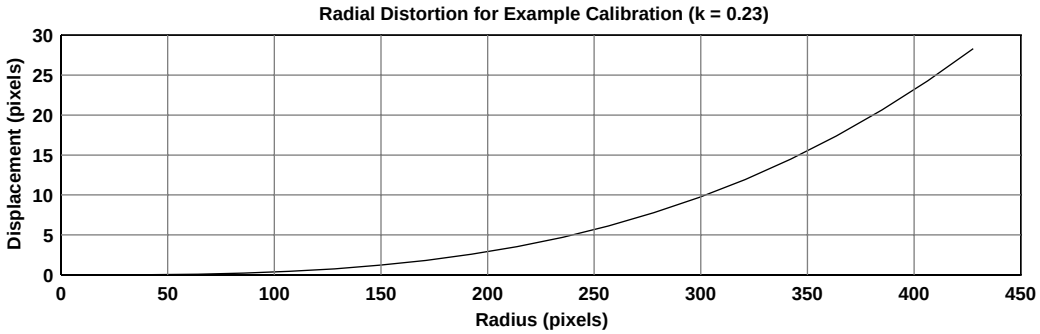


Figure 2.7 Plot of radial distortion for the example calibration.

2.7 Implementation

The camera calibration program for this project was implemented in ARM C++ to run on the Sun SPARCstation 10 with the X Window System. The program captures an image from two cameras and automatically performs the calibrations for each using the process described in this chapter. The screen capture in Figure 2.8 shows the window generated by the program which contains the results of two calibrations. The window shows the external and internal parameters of both cameras superimposed over each processed image.

When the camera calibration program ends, it saves the calibration data in a file for use by the stereo measurement program.

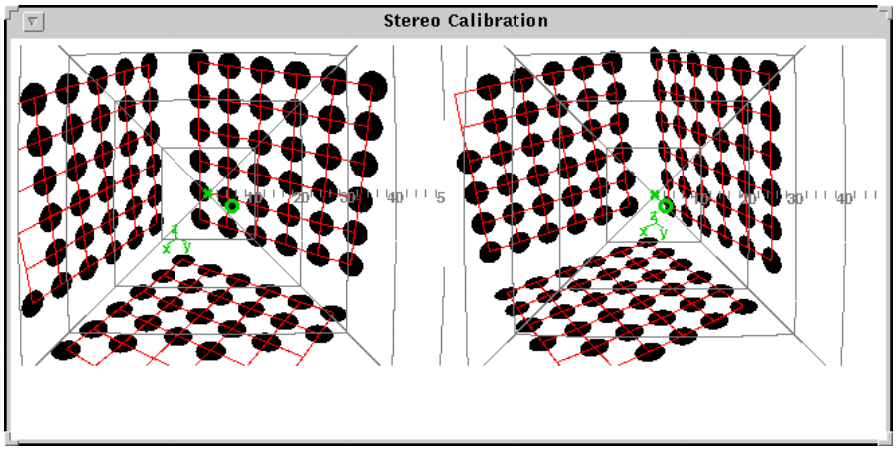


Figure 2.8 Screen capture of the camera calibration program.

Chapter 3

Stereo Measurements

3.1 Introduction

When a pair of calibrated cameras photograph a scene, the 3D positions of features in the scene can be measured from the resulting *stereo pair* of images. Measuring the position of a 3D point in the scene involves first locating two points, one from each image, which correspond to the 3D point. From those image points, lines-of-sight can be determined. The intersection of the lines results in a 3D measurement.

Normally, it is very difficult to find point correspondences in a pair of images for an unknown scene. For this project though, the problem is simpler because the scene consists of a known test pattern projected on the screen of a television. In other words, the system knows ahead of time what the features of interest are. As long as the same features are visible in both cameras, the system can use point ordering to obtain the matching features.

In this part of the project, the system uses stereo measurements to determine the pose of the television as well as measure the displayed test pattern. Using the knowledge of the television's pose, the system can transform the 3D geometry measurements into the television frame of reference as required by the measurement specifications. The depth information of the transformed points can then be thrown away to get an orthographic projection as seen from the front of the television.

3.2 Finding the Line-of-Sight

An initial step in taking a 3D point measurement from an image involves finding the line-of-sight for the point of interest in the image. To do this, use Equations (2.6) and (2.7) to convert the image point $\begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$ to the undistorted projection $\begin{bmatrix} P/S & Q/S \end{bmatrix}^{\mathrm{T}}$. The equation of the line-of-sight can be derived from Equation (2.1) to get

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}^{\mathrm{T}} \left(\lambda \begin{bmatrix} P/S \\ Q/S \\ 1 \end{bmatrix} - \mathbf{T} \right) \tag{3.1}$$

which is of the form

$$\mathbf{P} = \lambda \mathbf{dP} + \mathbf{P_0} \text{ where } \mathbf{dP} = \mathbf{R}^{\mathrm{T}} \begin{bmatrix} P/S \\ Q/S \\ 1 \end{bmatrix} \text{ and } \mathbf{P}_0 = -\mathbf{R}^{\mathrm{T}}\mathbf{T} \tag{3.2}$$

The 3D point in the scene lies on this parametric line. In a stereo system, there will be one line-of-sight for each camera.

3.3 Triangulation

With a line-of-sight from each camera, the intersection of the two lines yields a 3D position in the scene. To find the intersection, start with two parametric 3D lines.

$$\mathbf{P} = \lambda \mathbf{dP} + \mathbf{P_0} \tag{3.3a}$$

$$\mathbf{Q} = \eta \mathbf{dQ} + \mathbf{Q_0} \tag{3.3b}$$

Experimentally, the two lines-of-sight are not likely to intersect, but will instead pass close to one another. In a least squared error sense, the best intersection will occur on the line segment which connects to a point on each line and is perpendicular to both lines. The solution [28, 36] for the parameters which yields the end points of the line segment is,

$$\lambda = \frac{(\mathbf{b} \bullet \mathbf{dQ})(\mathbf{dP} \bullet \mathbf{dQ}) - (\mathbf{b} \bullet \mathbf{dP})(\mathbf{dQ} \bullet \mathbf{dQ})}{(\mathbf{dP} \bullet \mathbf{dQ})^2 - (\mathbf{dP} \bullet \mathbf{dP})(\mathbf{dQ} \bullet \mathbf{dQ})} \tag{3.4a}$$

$$\eta = \frac{(\mathbf{b} \bullet \mathbf{dQ})(\mathbf{dP} \bullet \mathbf{dP}) - (\mathbf{b} \bullet \mathbf{dP})(\mathbf{dP} \bullet \mathbf{dQ})}{(\mathbf{dP} \bullet \mathbf{dQ})^2 - (\mathbf{dP} \bullet \mathbf{dP})(\mathbf{dQ} \bullet \mathbf{dQ})} \tag{3.4b}$$

$$\text{where, } \mathbf{b} = \mathbf{Q}_0 - \mathbf{P}_0$$

provided that the lines do not point in the same direction. A good guess for the final answer is halfway between the points.

$$O = \frac{1}{2}\left(\lambda \mathbf{dP} + \mathbf{P}_0 + \eta \mathbf{dQ} + \mathbf{Q}_0\right) \tag{3.5}$$

This is a general way to take stereo measurements. An alternative and simplified solution results when the cameras face in the same direction [36].

3.4 Finding the Pose of the Television

Finding the orthographic projection of the television test pattern requires knowledge of the orientation of the television in the world's frame of reference. Since the system's main task involves measuring the distortions in the television's image, the system should not use a displayed pattern to find the television's pose. The system needs to find the orientation of the television screen independent of the television's ability to reproduce an image. One existing method [11] uses knowledge of the television's cabinet shape to find the orientation. Since the implementation in this project needs to operate on any television without knowing its properties beforehand, the implementation uses a compromise. By displaying a flat white field on the screen (shown in Figure 3.1) the program can automatically find the corners of the screen, their 3D world positions, and consequently the orientation of the screen. This trick presumes that the flat white field fills the active area of the screen, that the displayed edges are a function of the mechanical construction of the screen, and that the corners of the screen fall in a plane tangent to the screen. These are reasonable assumptions.

To find the pose of the television, the automated procedure first captures a stereo image pair from the cameras while the television displays a flat white field. The procedure then takes one of the images and

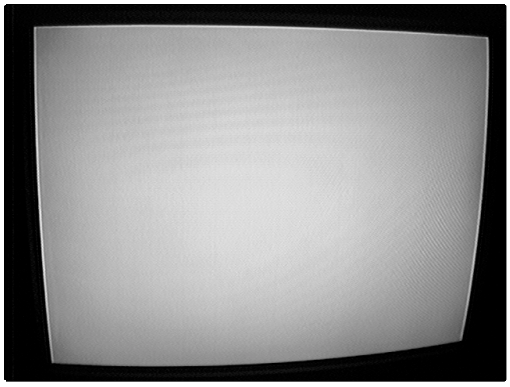


Figure 3.1 Unaltered video image of a flat white field.

looks at a corner of it about 1/5 the total image width. Using the smaller image, the procedure computes the sum of the absolute values of the horizontal and vertical image gradients, yielding the edge image in Figure 3.2 (a). A gradient of Gaussian filter was used in the horizontal and vertical directions to both filter the image and find the edges. The procedure then fits a line to the near-vertical edge and to the near-horizontal edge of the white field by sampling two points on each edge as an initial guess. Figure 3.2 (b) shows such an initial guess. Next, a nonlinear search refines each fit based on the criterion of maximizing the sum of the gradient along each line.

During the nonlinear search, the idea is to position the line estimate along the peaks of the image gradient which exist along an image edge. Since a two-dimensional line has only two degrees of freedom, it can be represented by the following equation.

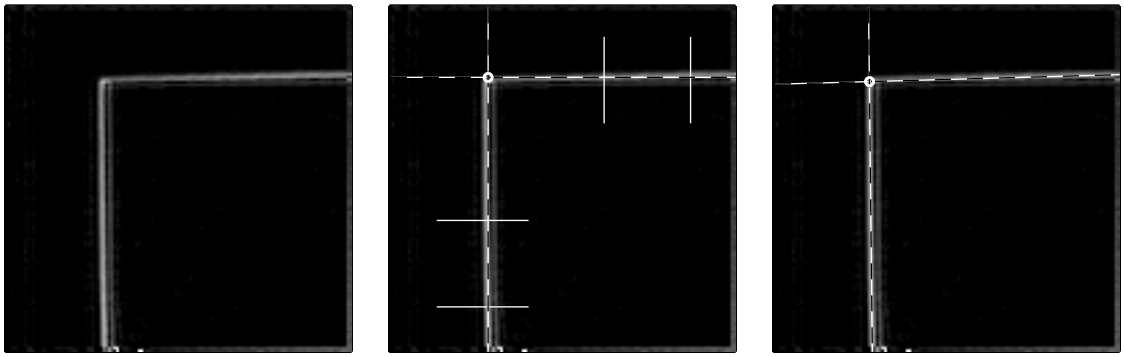
$$x \cdot l_x + y \cdot l_y + l_z = 0 \text{ where } l(r,\theta) = \begin{bmatrix} \cos\theta \\ \sin\theta \\ -r \end{bmatrix} \tag{3.6}$$

Given the previous representation, optimize Equation (3.7).

$$\max \varepsilon(r,\theta) = \begin{cases} \sum_{x=0}^{w-1} \mathbf{I}\left(x, \left\lfloor -\left(l_z + x \cdot l_x\right)/l_y + 0.5 \right\rfloor\right) & \text{if } |l_y| > |l_x| \\ \sum_{y=0}^{h-1} \mathbf{I}\left(\left\lfloor -\left(l_z + y \cdot l_y\right)/l_x + 0.5 \right\rfloor, y\right) & \text{otherwise} \end{cases} \tag{3.7}$$

The solution to Equation (3.7) is a vector $[r,\theta]$ that maximizes ε. Again, the downhill simplex algorithm [31] can be used to perform the search. Because the downhill simplex method is a minimizer, the system actually minimizes Equation (3.7) for $-\varepsilon$. Figure 3.2 (c) shows the results of an example nonlinear search.

Finally, the procedure interprets the intersection of the two lines as a corner of the screen. This process repeats for all corners in both images.



(a) (b) (c)

Figure 3.2 Finding a corner of the flat white field: (a) edge image; (b) initial guess; (c) optimized estimate.

Using the measured positions of the corners in each image, the procedure can now find the pose of the television. The procedure does this by first finding the lines-of-sight for each corner. Then for each corresponding corner pair, it triangulates to find the 3D position of each screen corner.

With the 3D positions of the four corners $\mathbf{P}_0$ - $\mathbf{P}_3$ (in clockwise order), Equation (3.8) gives the position of the center of the television screen in the world frame, while Equation (3.9) gives the rotation of the world frame to the television's frame of reference.

$$\mathbf{T}_{TV} = \frac{1}{4}\left(\mathbf{P}_0 + \mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3\right) \tag{3.8}$$

$$\mathbf{R}_{World-TV} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3\end{bmatrix}^{\mathrm{T}} \tag{3.9}$$

where $\mathbf{b}_1 = \left(\mathbf{P}_1 - \mathbf{P}_0\right)/\left\|\mathbf{P}_1 - \mathbf{P}_0\right\|$, $\mathbf{b}_2 = \left(\mathbf{P}_3 - \mathbf{P}_0\right)/\left\|\mathbf{P}_3 - \mathbf{P}_0\right\|$, and $\mathbf{b}_3 = \mathbf{b}_1 \times \mathbf{b}_2$

Equation (3.10) provides the transformation from the world frame to the television frame.

$$\mathbf{P}_{TV} = \mathbf{R}_{World-TV}\left(\mathbf{P}_{World} - \mathbf{T}_{TV}\right) \tag{3.10}$$

Now that the pose of the television is known, the measurements on the test pattern can be made.

3.5 Finding the Features of the Crosshatch Image

The measurements of the geometric distortion in the television are made using a standard crosshatch pattern like the one shown in Figure 3.3. The measurements are taken at the crosshatch intersections. A procedure for automatically finding and measuring the intersections follows.

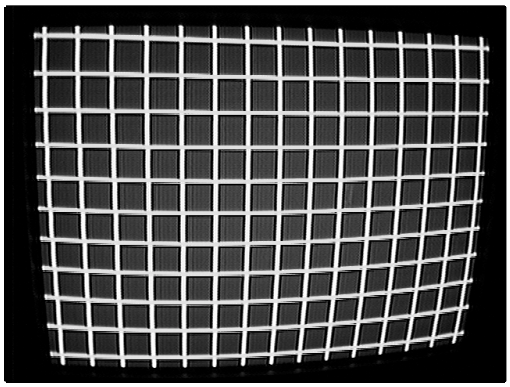


Figure 3.3 Unaltered video image of a crosshatch.

The first step in finding the intersections of the crosshatch involves thresholding and inverting the crosshatch image. The resulting binary image is combined with a thresholded version of the flat white field to yield the image in Figure 3.4 (a). A morphological closing followed by an opening are performed to clean up the image [35]. This operation tends to fill in small holes and remove small specks from the image. The result is also skeletonized [35] as shown in Figure 3.4 (b).

Combining the thresholded crosshatch and its skeleton produces the image in Figure 3.4 (c). The procedure uses this image later on as a mask. The procedure then performs a connected component analysis [35] on the mask. The connected component analysis assigns unique labels to the black regions of the image. This will help identify the crosshatch intersections later on. In the final image processing step, the procedure finds the sum of the absolute values of the horizontal and vertical gradients of the crosshatch image to yield the edge image in Figure 3.4 (d).

As can be seen, the edge image of the crosshatch consists of many closed boxes. In order to obtain the image positions of the crosshatch intersections, the automated procedure tries to find the corners of each box by looking at each labeled region. The corners of the boxes can then be combined to get the intersection. In order to know which boxes are adjacent to one another, the region centroids shown in Figure 3.4 (c) are sorted into rows and columns.

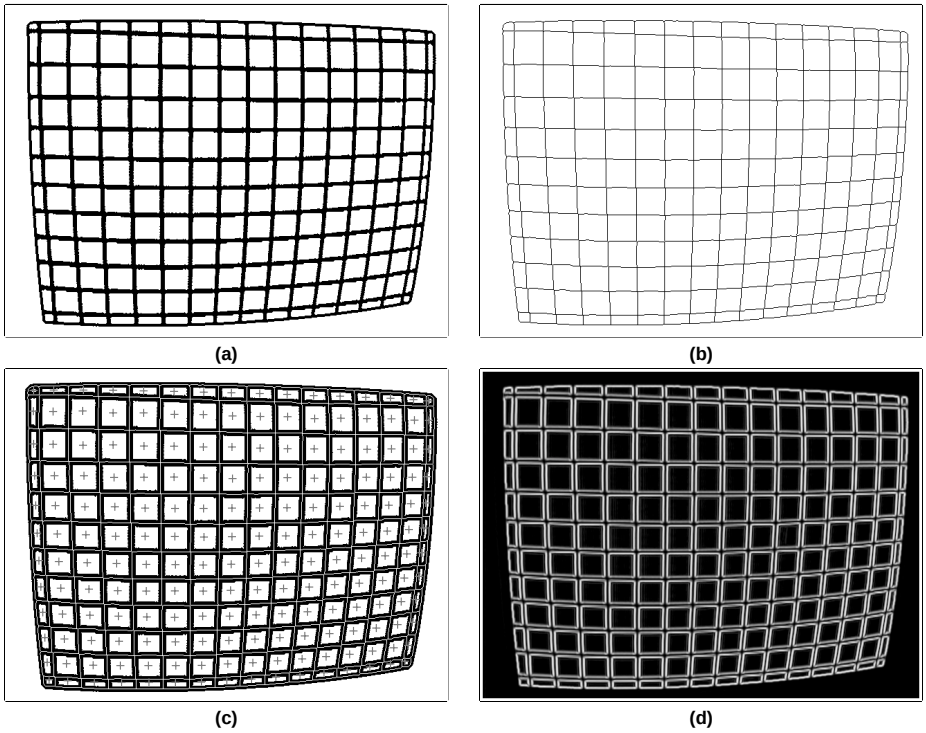


Figure 3.4 Image processing for the crosshatch measurement procedure: (a) thresholded image; (b) skeleton of thresholded image; (c) mask image with labeled features; (d) edge image.

Figure 3.5 (a) shows the initial label numbers for the mask image resulting from the connected component analysis. The program sorts the boxes by locating the nearest neighbors of each region in both the horizontal and vertical directions. With the knowledge of the nearest horizontal and vertical neighbors the program can "walk" across the rows and columns to sort the labels. The first step in sorting the labels into rows and columns involves finding the top-left label. The program does this by starting from the first label and walking up until it can find no more points and then walking left until it reaches the end. With the top-left point known, the procedure can walk each row and assign new, sorted labels to the regions. Figure 3.5 (a) shows an example of the walk, while Figure 3.5 (b) shows the resulting sorted labels.

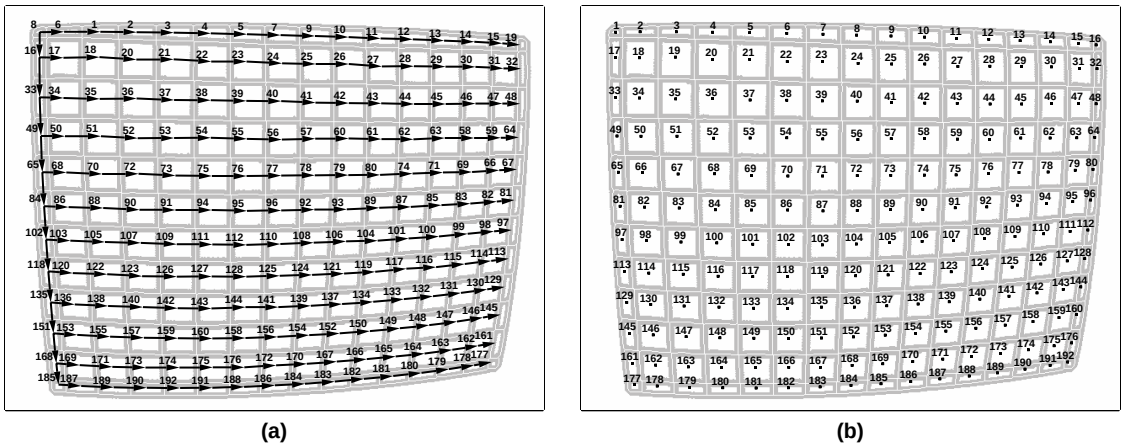


Figure 3.5 Sorting the grid regions: (a) initial labels and walk; (b) sorted labels.

With the boxes are sorted into rows and columns, the procedure then measures the image position of each box corner. For each labeled region, an initial guess is found for a box which fits the region. The procedure finds the initial guess by looking for the four corners of the box. It uses a radial search over one quadrant starting from the region centroid and looking outward for a maximum in the gradient magnitude. The labeled mask image defines the boundary for the radial search. Figure 3.6 (a) shows an example radial search in the top-left quadrant. The box corner is estimated from the discontinuity in the polar radial distances. In this case the point with the maximum radial distances is chosen as the corner. Figure 3.6 (b) shows a graph of radial distance versus angle for the quadrant search. The dashed line indicates the point chosen as the corner. Figure 3.6 (c) shows the resulting initial guess for the top-left corner.

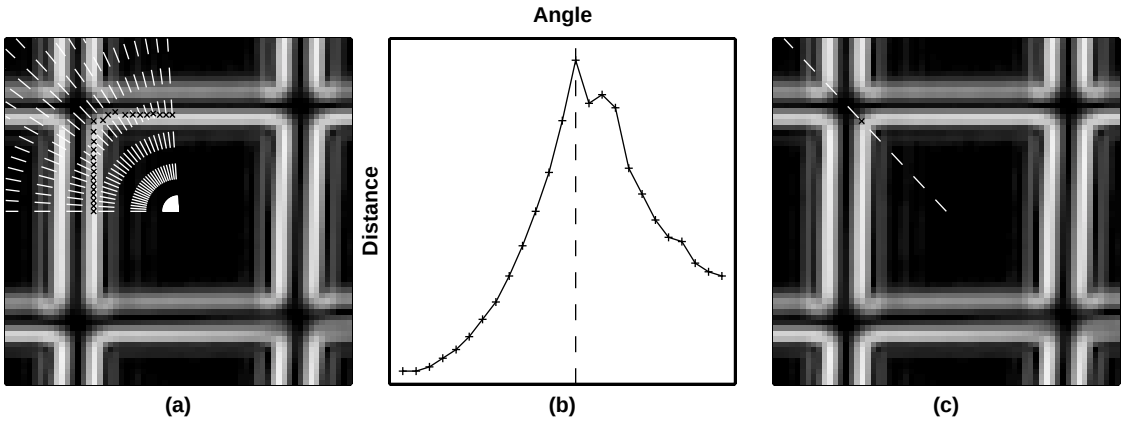


Figure 3.6 Finding an initial guess for a corner of a crosshatch box: (a) polar quadrant search; (b) plot of distance from center to edge versus angle; (c) resulting guess.

The procedure searches all four quadrants to get an initial guess for the four corners of the box. Figure 3.7 (b) depicts a possible initial guess for the box. A nonlinear search then refines the fit by trying to maximize the sum of the absolute values of the gradient along the four line segments of the box. This is similar to the corner locating procedure used on the flat white field. In this case, the search is 8 dimensional. The nonlinear search tries to find a vector $[x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4]$ which maximizes the sum of the gradients. The search uses the labeled mask image to constrain the search in order to prevent the search from "spilling over" into an adjacent region. Figure 3.7 (c) shows the result of the nonlinear search. The procedure repeats this process for each labeled region in each image of the stereo pair.

The next step is to average together the box corners surrounding a crosshatch intersection and store the result as the center of the respective crosshatch intersection. Figure 3.8 shows an example of four corners averaged into a resulting intersection measurement.

Since the labeled mask regions are sorted into columns and rows, the crosshatch intersections end up being sorted as well. As a result, the corresponding intersections in each image are known.

Finally, by using the corresponding 2D crosshatch intersections, the triangulations are found and the resulting 3D locations of the intersections are transformed into the TV frame of reference using Equation (3.10). Discarding the depth information results in an orthographic projection of the test pattern.

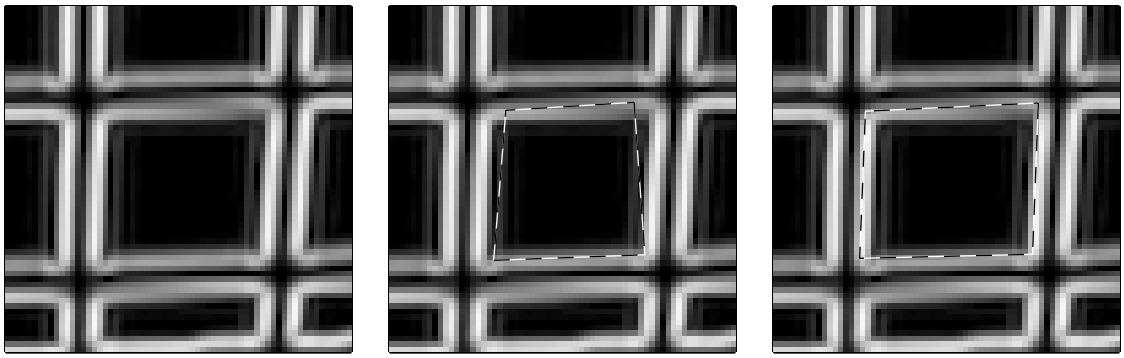


Figure 3.7 Search for edge corners: (a) edge image; (b) initial guess; (c) optimized fit.

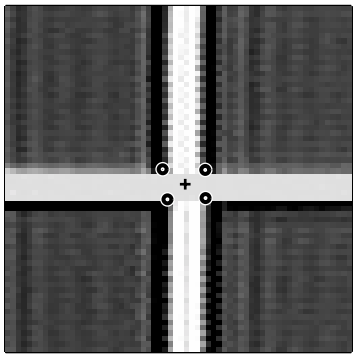


Figure 3.8 Final two-dimensional measurement of a grid intersection: the circles are the measured corners and the cross is the average of the four corners.

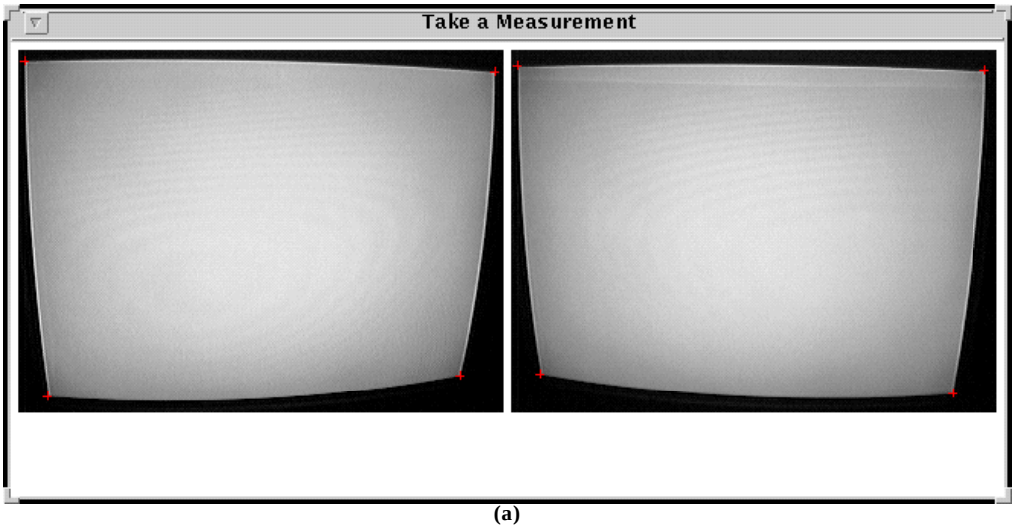
3.6 Implementation

The stereo measurement program for this project was implemented in ARM C++ to run on the Sun SPARCstation 10 with the X Window System. The program has two parts: the pose estimation part and the crosshatch measurement part.

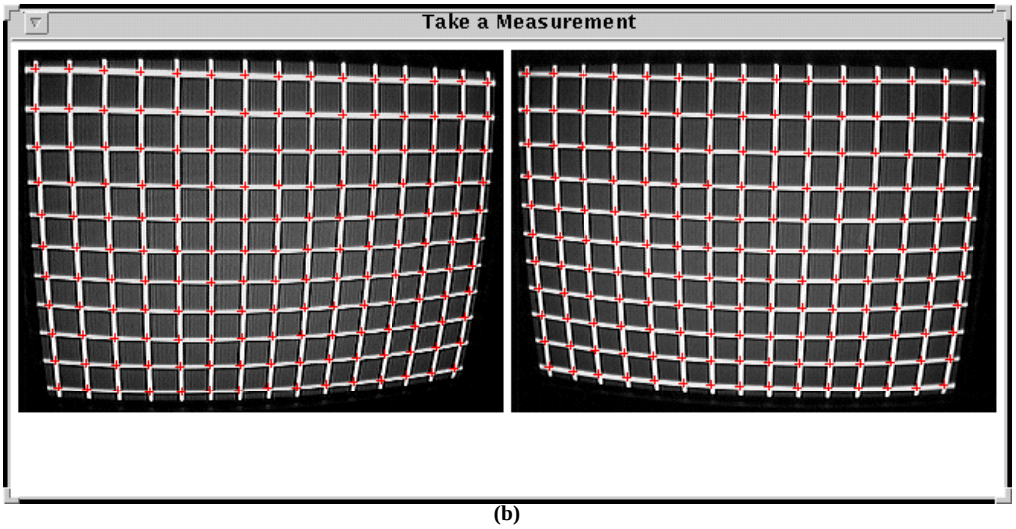
The pose estimation part of the program captures an image from the two cameras and automatically performs the pose estimation for the television using the process described in this chapter. The screen capture in Figure 3.9 (a) shows the window generated by the program which displays the results of the corner finding.

The crosshatch measurement part of the program captures a second pair of images from the cameras and automatically performs the stereo measurements on the test pattern displayed on the television. The screen shot in Figure 3.9 (b) shows the window generated by the program as it displays the results of the intersection finding.

When the stereo measurement program ends, it saves the positions of the crosshatch intersections in a file.



(a)



(b)

Figure 3.9 Screen captures of the stereo measurement program: (a) finding the television's pose; (b) finding the crosshatch intersections.

Chapter 4

Results and Conclusions

4.1 Experimental Results

In order to test the automatic display inspection system described in this thesis, the system was used to measure the geometric distortions present in a Philips/Magnavox 32 inch color CRT television. A standard 13x17 crosshatch pattern was displayed on the television and measurements were taken for 9 different camera views. An Apple Power Macintosh 6100/60 AV with composite video output, generated the crosshatch and fed the composite video input of the television-under-test. Figure 4.1 depicts the straight-on orthographic projections of the resulting measured crosshatch intersections. In the figure, the measurements from the different trials are overlapped showing consistent results. The gray dashed grid is a best-fit rectangular crosshatch for the data. Geometric deviations in the display are noticeable, especially on the top row of intersections. The deviations were visually confirmed to be present in the television display.

Figure 4.2 shows the range of deviations of the measurements. The measurements for each crosshatch intersection are plotted about their mean for all of the trials. All of the points fall within a radius of 1/10 of an inch with a standard deviation of 0.02 inches.

The plots of Figure 4.3 are enlargements of the four outer edges of the crosshatch measurements. They show the extent of the outline distortion in the display. The top edge, for instance, has a large degree of pincushioning.

The plots in Figure 4.4 show the non-linearity in the television display. The non-linearity was calculated using Equations (1.3).

The various figures show that for any single trial, there are errors in the measurements. When taken together though, the results from multiple trials follow consistent trends. This indicates that multiple measurements taken on the same television could be combined to obtain more accurate results.

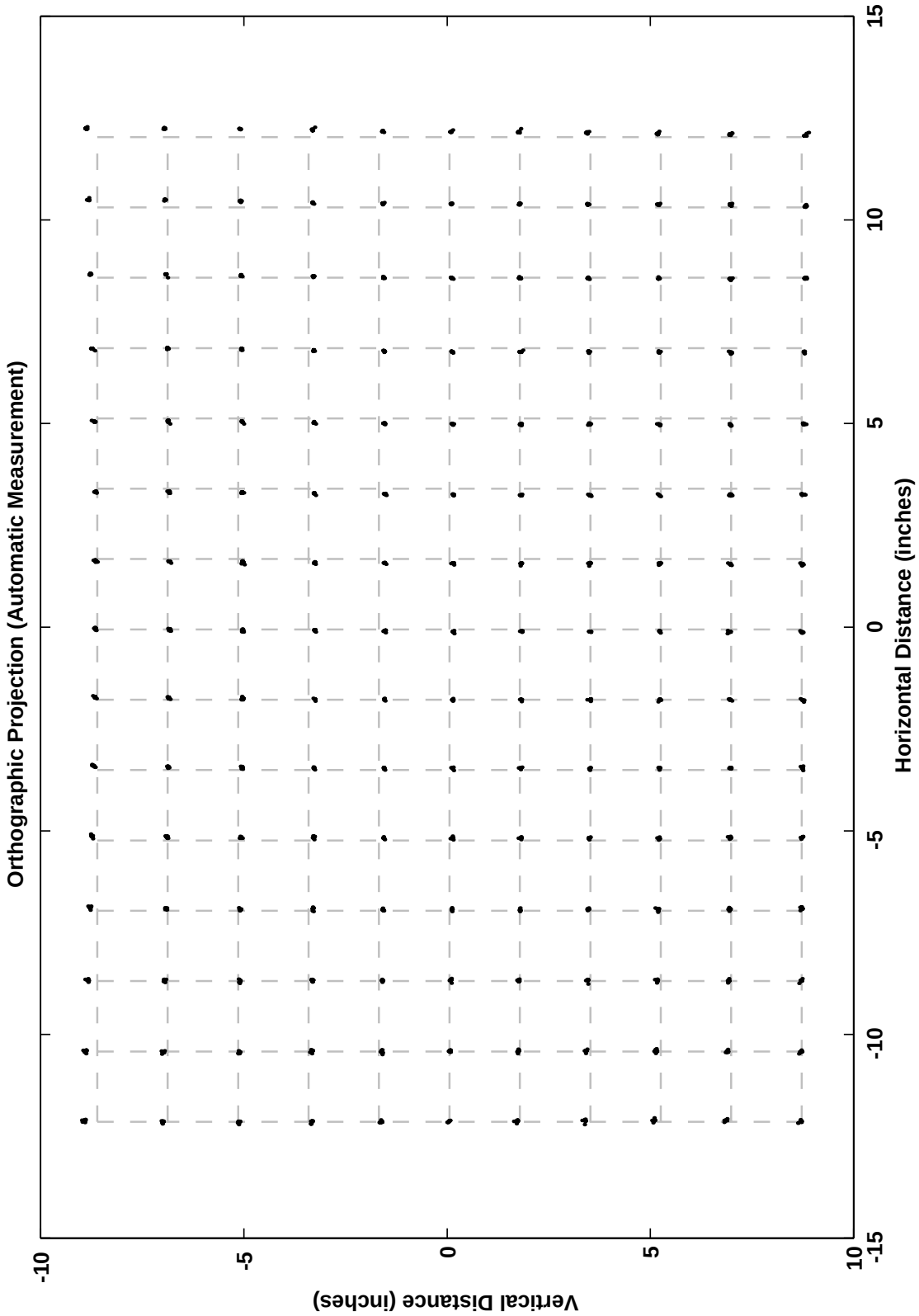


Figure 4.1 Plot of test measurements performed by the automatic geometry measurement system.

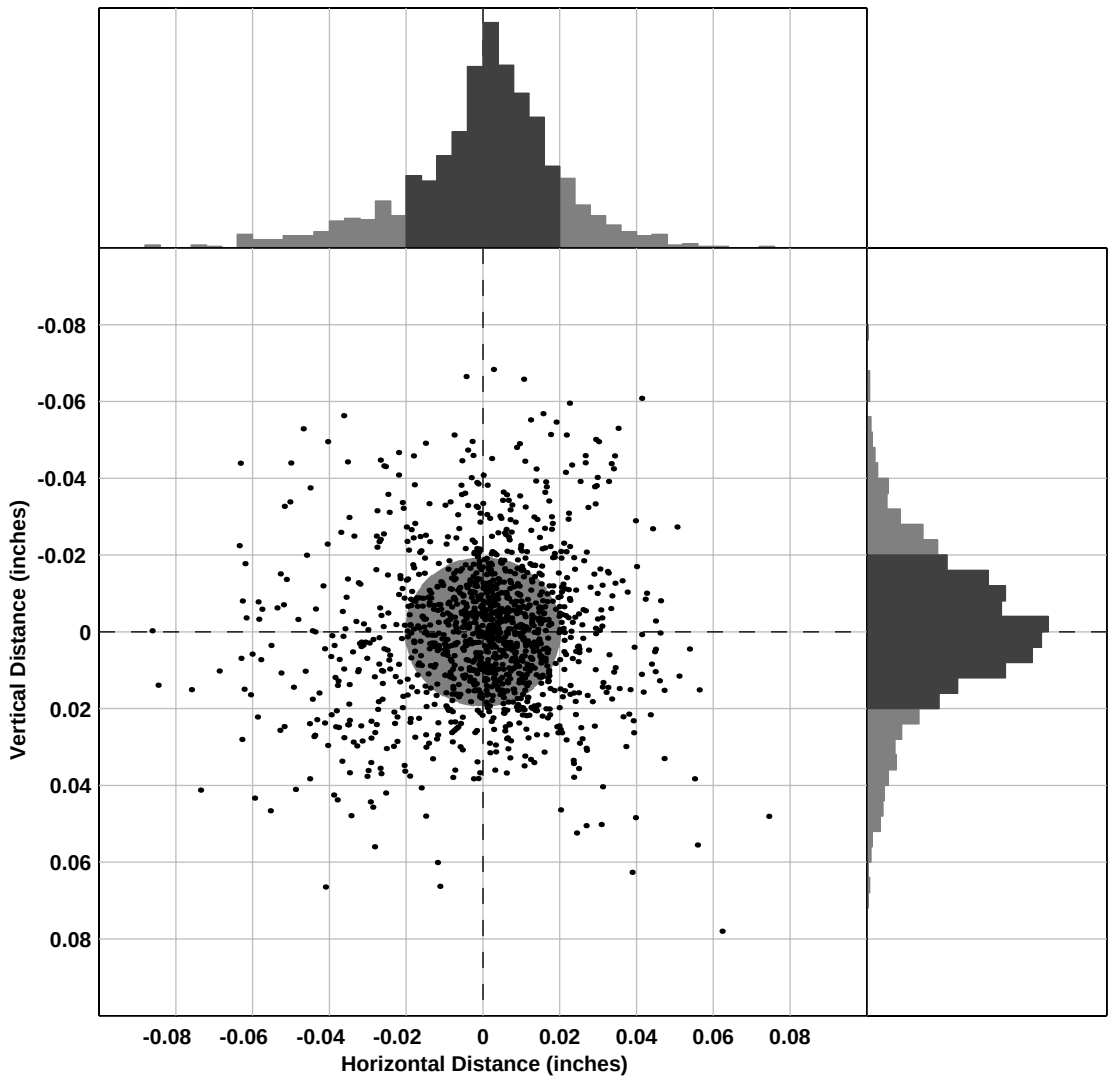


Figure 4.2 Plot of deviations in measurements from their mean (the gray circle depicts a standard deviation of 0.02 inches).

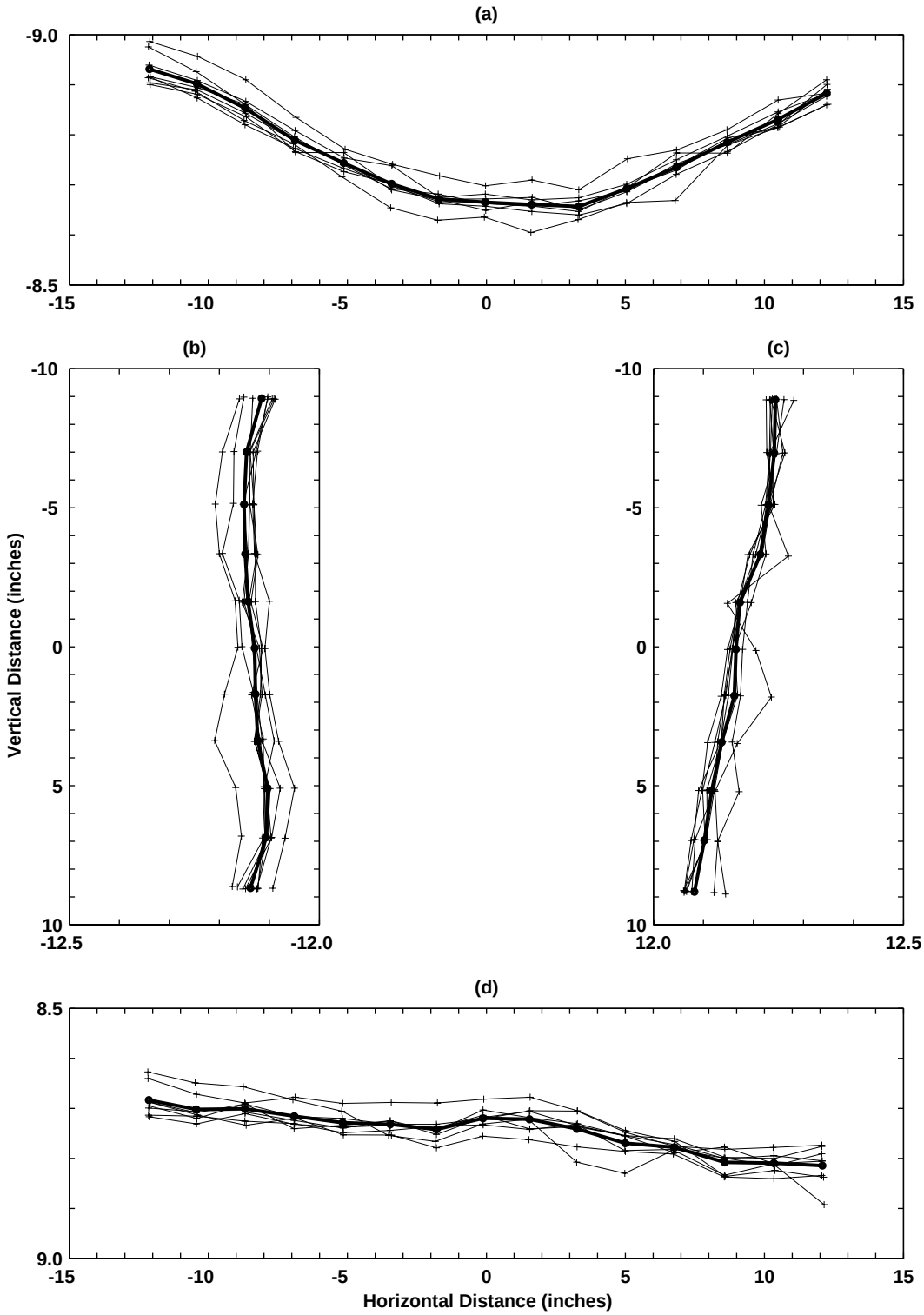


Figure 4.3 Enlargement of the outline distortions in the TV image: (a) top edge; (b) left edge; (c) right edge; (d) bottom edge.

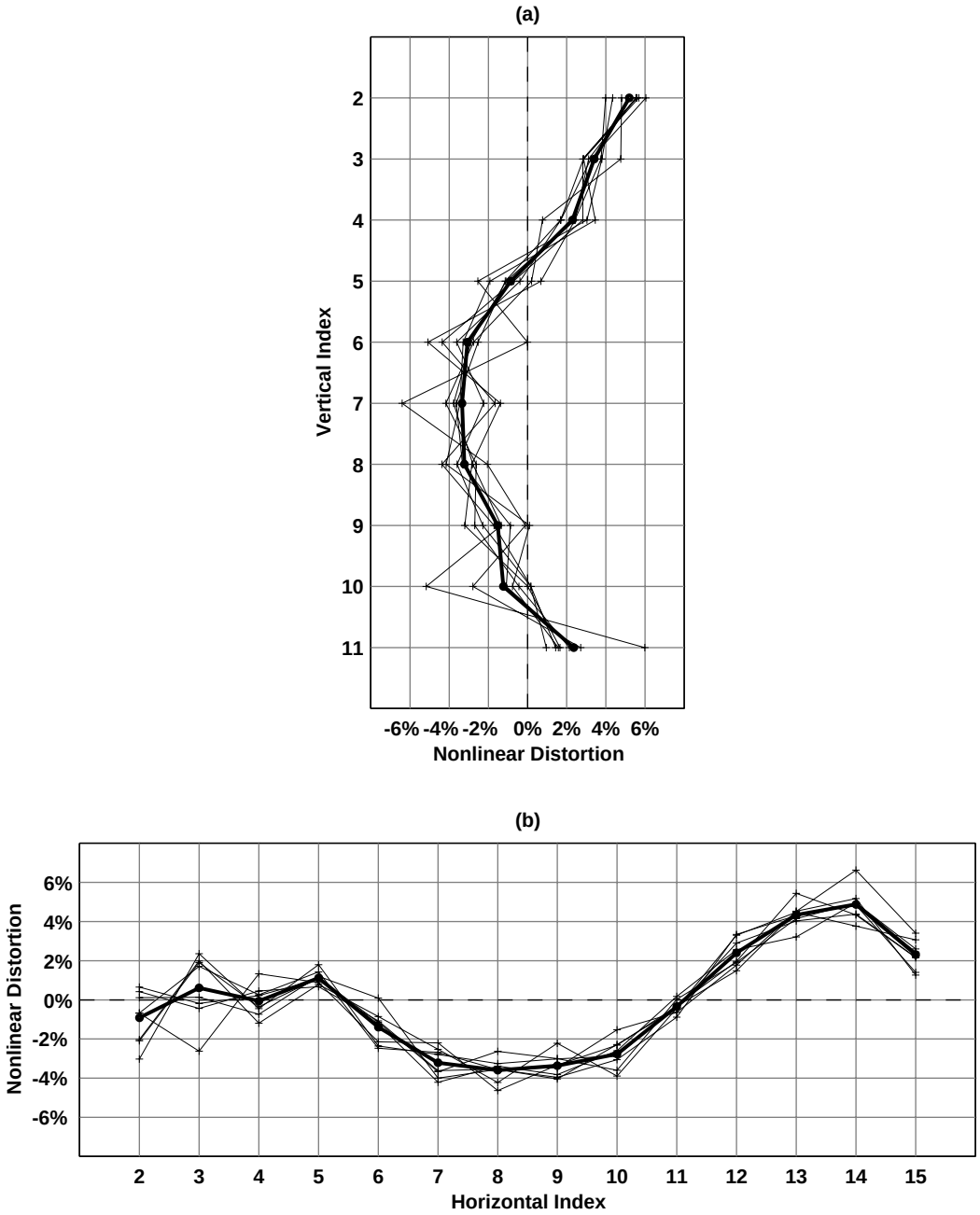


Figure 4.4 Non-linearity measurements on the TV image: (a) vertical non-linearity; (b) horizontal non-linearity.

4.2 Future Work

There are a number of improvements that could be made to this automatic display inspection system. A list of possible future work follows.

4.2.1 Off-Line Camera Calibration

Since this is a laboratory based inspection system, the internal parameters of the cameras could be calibrated off-line. Consequently, a more detailed distortion model could be used resulting in more accuracy. This would also speed up the external calibration of the cameras when they are placed in front of the TV. Separating the model into external and internal parameters would involve improving the camera model though. In practice, the model used in this project, which is fairly commonplace in the literature, works best when both the external and internal parameters are searched for simultaneously. Changing the model would allow the calibration to be broken up into simpler stages where only a few parameters are found at a time. There is much prior work done in this area, but it is not such a popular approach.

Furthermore, the cameras could be placed on a rigid stereo rig. This would allow the external parameters to be precalculated as well, eliminating the camera calibration step altogether during a measurement.

4.2.2 Correction for Refraction

A second improvement to this inspection system would be to correct for the refraction caused by the CRT glass. A typical CRT face consists of a curved, leaded glass plate about 1/2 inch thick with the image phosphors painted on the interior face. Due to refraction in the face, the perceived image is a function of viewing position. Figure 4.5 shows a simulated cross-section of a CRT faceplate refracting the lines-of-sight from two cameras.

The figure shows that the perceived intersection of the lines does not coincide with the actual physical position of the image on the phosphor. The perceived position can move with the positions of the cameras resulting in parallax errors. Nonetheless, the orthographic projections of the perceived intersection and the actual intersection are very close (within a fraction of a millimeter). In other words, by using two cameras the resulting measurement resembles the desired measurement.

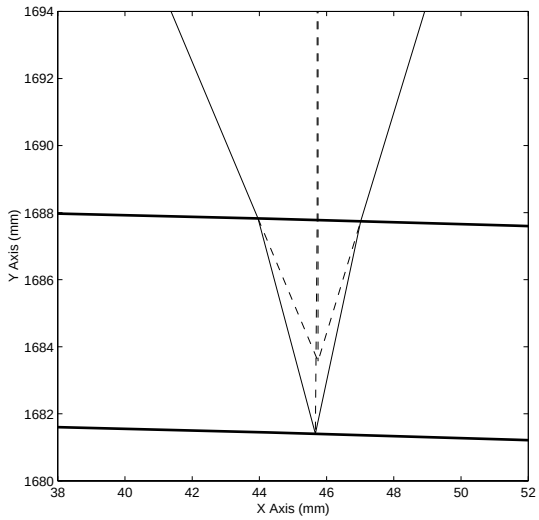


Figure 4.5 A simulated plot of the refraction due to glass thickness; the perceived intersection of the lines of sight falls short of the actual intersection.

It does not appear that glass thickness, shape, and index of refraction can be measured with a stereo pair of cameras without a-priori information. This is probably why the only approach that tries to account for refraction uses manufacturing information about the CRT. However, it seems that more than two views could be used to determine the parameters of the CRT.

4.2.3 Faster Execution

A third drawback to the inspection system is slow execution. Using a SPARCstation 10 with 128 MB of RAM running the SunOS 5.5 operating system, it takes as much as 18 minutes to perform the measurements on one television including setup time. Computationally, the camera calibration takes about 30 seconds for both cameras, the TV pose estimation takes about 15 seconds, while the stereo measurements take roughly 8 minutes. Real world display inspection systems are much faster. In order to speed up the calculations, some of the algorithms used could be simplified. For example, the algorithms in the measurement stage which perform a great deal of brute-force nonlinear searches in order to obtain sub-pixel measurements could use some optimization.

4.3 Conclusions

Before the advent of automatic television image inspection systems, display manufacturing and testing was a very laborious process which required manual skilled labor. As computers became small, fast and commonplace, automatic inspection approaches slowly emerged. They started out simple and later developed into rather involved computer vision systems. As modern solid-state technology threatens the CRT display, it is modern technology that has breathed new life into the CRT market. Automatic inspection and adjustment has made the CRT display higher quality and cheaper staving off its supposed inevitable demise.

While there are many approaches used in automatic display inspection, they seem to fall into two major groups:

The *fixtured systems* involve attaching the camera sensors to a rig which fits to the display-under-test. Consequently, a fixtured system knows the pose of the cameras relative to the television. Furthermore, the fixtured approaches fall in to two subgroups; *fixed camera* and *movable camera*. When the cameras are fixed they can individually focus on parts of the display or they can view the entire display resulting in stereo views. On the other hand, a robotically movable camera usually focuses on a small part of the display. Because the camera is close to the display and the position of the camera is known, problems like parallax do not pose a problem. Also, because of the close proximity of the camera to the CRT, a movable camera system can measure properties like color convergence and beam focus. Ultimately though, fixtured approaches are not too practical for assembly line work. Fixtured approaches occur more in the ITC adjustment process which requires mechanical contact with the CRT.

The *position independent* systems allow the cameras to be placed freely in front of the display-under-test as long as the cameras have a good view of the screen. This approach has two subgroups as well; *single camera* and *multiple camera*. The single camera approach uses a-priori knowledge of the mechanical shape of the display to find the pose of the camera and recover 3D measurements. Alternatively, the multiple camera approach (usually with two cameras) does not require any knowledge of the shape of the television in order to make usable measurements.

It is noteworthy that all of the common approaches to computer vision based automatic display inspection are covered by patents. This commercially protects the approaches used by the major players in the field by giving them a limited monopoly in exchange for a detailed description of their inventions. Admittedly, the market for automatic inspection/adjustment systems is limited and, rightly so, very competitive. There is very little market for laboratory inspection systems. The major market is assembly line inspection/adjustment systems. These systems typically have either a high cost or are leased to display manufacturers.

Bibliography

Bibliography

Display Inspection/Alignment

SID - Society for Information Display

[1] *IEC Final Draft Intl. Std. 100A/5/FDIS, Methods of Measurement on Receivers for Television Broadcast Transmissions - Part 1: General Considerations - Measurements on Displays at Radio and Video Frequencies*. IEC, 1996.

[2] Robert Fawcett. "Stereo Imaging and Automated Vision Boost CRT Production." *Photonics Spectra* 29, no. 3 (March 1995): 92-94.

[3] John Melson. "CRT and Flat Panel Inspection Via PC-Based Image Processing." *Proc. of SPIE--The International Society for Optical Engineering* 974 (August 1988): 151-56.

[4] S. Uno, R. Douji, M. Inoue, and Y. Goto. "Automatic Evaluation System for CPT Picture Characteristics." *Proc. of IEEE IECON* (1984): 432-37.

[5] Robert Lin. "Automated CRT Inspection and Alignment." *Information Display* 4, no. 6 (June 1988): 16-17.

[6] Greg A. Kern. "CRT-Display Inspection with a Solid-State Camera: 3D Modeling and Parallax Correction." *SID International Symposium* 25 (June 1994): 791-94.

[7] Eric Buckley, Mark Kao, and Bruce Lee. "Technology Options for Improved ITC Test and Alignment." *SID Display Manufacturing Technology Conference* 2 (January 1995): 33-34.

[8] T. Hibara, and M. Hayashi. "Automatic Adjustment for Color Display Monitor." *Proc. of IEEE IECON* (1986): 164-69.

[9] George J. Yang. "A Practical Automatic Alignment and Inspection System for Monitor Manufacturers." *SID International Symposium* 24 (May 1993): 10-13.

[10] James R. Webb. "Microchip Architecture for Full Digital Control of Geometry, Convergence, and Colorimetry in CRT Monitors." *SID International Symposium* 25 (June 1994): 7-10.

[11] James R. Webb, Ron C. Simpson, and Howard M. Pogoda. *Automatic Precision Video Monitor Alignment System*. US patent 5,216,504 to Display Laboratories, Inc. Washington, DC: Patent and Trademark Office, 1993.

[12] IBM Corp. "Automatic Measurement of Display Test Pattern." *IBM Technical Disclosure Bulletin* 27, no. 12 (May 1985): 7203-4.

[13] Akira Nakamura, Tatsunori Hibara, Naomichi Yamada, and Satoshi Hori. "Display Monitor Intelligent Adjusting System: DIAS." *Proc. of IEEE IECON* (1989): 715-20.

[14] Eric Buckley, Branko Bukal, Wayne Dawe, Paul Farrer, Karoly G. Nemeth, and Andrew Noonan. *Test and Alignment System for Electronic Display Devices and Test Fixture for Same*. US patent 5,969,756 to Image Processing Systems Inc. Washington, DC: Patent and Trademark Office, 1999.

[15] Jun Mochizuki, Toshio Asano, Kinuyo Hagimae, Teruo Matsuo, and Hiryouki Yoshida. "Adjustment Support and Inspection System for Large Color CRTs with Novel Picture Quality Sensors." *SID Display Manufacturing Technology Conference* 3 (February 1996): 49-50.

[16] Mike Lindahl, Zaher Samman, and Eric Beekers. "A Factory Automated Projection TV Digital Convergence System." *IEEE Int. Conf. on Consumer Electronics* 40, no. 3 (1994): 348-49.

[17] Leonard I. Suckle. "Automatic Alignment Techniques for Color Television Manufacturing." *IEEE Trans. on Consumer Electronics* 34, no. 4 (November 1988): 886-93.

[18] Charles Chuang, Warren Chen, Ted Wang, Jackson Wen, Anthony Yeh, Teresa Hsieh, James R. Webb, Greg Kern, and Seven J. Lassman. "A Non-impact High Resolution Geometry Alignment System for Monitor Production." *SID Display Manufacturing Technology Conference* 1 (January 1994): 12-13.

[19] Yi-Ping Hung, Yau-Liang Tsai, and George J. Yang. *Method and a System for Testing a Cathode Ray Tube or Like Products*. US patent 5,442,391 to EeRise Corp. Washington, DC: Patent and Trademark Office, 1995.

[20] James E. Thario, and James R. Webb. *Correction for Monitor Refraction using Empirically Derived Data*. US patent 5,657,079 to Display Laboratories Inc. Washington, DC: Patent and Trademark Office, 1997.

[21] James R. Webb, and Gregory A. Kern. *Method and Apparatus for Transforming Coordinate Systems in an Automated Video Monitor Alignment System*. US patent 5,510,833 to Display Laboratories Inc. Washington, DC: Patent and Trademark Office, 1996.

[22] Yi-Ping Hung, Dong-Yueh Liu, Yau-Liang Tsai, and George J. Yang. "A New High Performance Automatic Geometry Alignment System for Monitor Manufacturing." *SID Display Manufacturing Technology Conference* 2 (January 1995): 37-38.

[23] David A. Fridge. *Stereoscopic Electro-Optical System for Automated Inspection and/or Alignment of Imaging Devices on a Production Assembly Line*. US patent 5,638,461 to Kollmorgen Instrument Corp. Washington, DC: Patent and Trademark Office, 1997.

[24] Arjun Ramamurthy, and Alan Fridge. "Distortion Measurement and Simultaneous Adjustment of Multiple Interacting Controls on CRT Monitors." *SID Display Manufacturing Technology Conference* 3 (January 1996): 55-57.

[25] Roger Y. Tsai. "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses." *IEEE Journal of Robotics and Automation* RA-3, no. 4 (August 1987): 323-44.

[26] Juyang Weng, Paul Cohen, and Marc Herniou. "Camera Calibration with Distortion Models and Accuracy Evaluation." *IEEE Trans. on Pattern Analysis and Machine Intelligence* 14, no. 10 (October 1992): 965-80.

[27] Hanqi Zhuang, and Zvi S. Roth. *Camera-Aided Robot Calibration.* Boca Raton: CRC Press, 1996.

[28] K. B. Atkinson. *Close Range Photogrammetry and Machine Vision*. Scotland, UK: Whittles Publishing, 1996.

[29] Duane C. Brown. "Decentering Distortion of Lenses." *Photogrammetric Engineering* 32, no. 3 (1966): 444-62.

[30] Reimar K. Lenz, and Roger Y. Tsai. "Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3D Machine Vision Metrology." *Proc. of IEEE Int. Conf. on Robotics and Automation* (1987): 68-75.

[31] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. 2d ed. Cambridge: Cambridge Univ. Press, 1992.

[32] Y. I. Abdel-Aziz, and H. M. Karara. "Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry." *Symposium on Close-Range Photogrammetry*. Urbana, Illinois (January 1971): 1-18.

[33] William I. Grosky, and Louis A. Tamburino. "A Unified Approach to the Linear Camera Calibration Problem." *IEEE Trans. on Pattern Analysis and Machine Intelligence* 12, no. 7 (July 1990): 663-71.

[34] Sundaram Ganapathy. "Decomposition of Transformation Matrices for Robot Vision." *Proc. of IEEE Int. Conf. on Robotics and Automation* (March 1984): 130-9.

[35] Rafael Gonzalez. *Digital Image Processing*. Reading, Mass.: Addison-Wesley, 1992.

[36] Robert Haralick, and Linda Shapiro. *Computer and Robot Vision*. Vol. 2. Reading, Mass.: Addison-Wesley, 1993.

Appendix

Appendix

Selected Parts of the Implementation

Below is a list of the files written for the ARM C++ implementation of the Camera Calibration program and the Stereo Measurement program. The programs were built using the Sun C++ 4.1 compiler. The linear and nonlinear optimizations were performed with the Numerical Recipes [31] library, video capture was performed with the Solaris XIL 1.1 Imaging Library, and windowing was performed with X Windows.

Geomess
- Autocalibrate.cc
- Autocalibrate.h
- DoCalibrate.cc
- DoMeasure.cc
- FindCorners.cc
- FindGrid.cc
- SortGrid.cc
- SortGrid.h

ImageLib
- Image.cc
- Image.h
- ImageConvolve.cc
- ImageConvolve.h
- ImageCopy.cc
- ImageCopy.h
- ImageEdges.cc
- ImageEdges.h
- ImageFile.cc
- ImageFile.h
- ImageHough.cc
- ImageHough.h
- ImageLabel.cc
- ImageLabel.h
- ImageLabelUtils.cc
- ImageMorphology.cc
- ImageMorphology.h
- ImagePointProcess.cc
- ImagePointProcess.h
- PGMUtils.cc
- PGMUtils.h

VisionLib
- calibrate.h
- calibrateLinear.cc
- calibrateNonlinear.cc
- calibratePlumbLine.cc
- calibrateTsai.cc
- camera.cc
- camera.h
- lines.cc
- lines.h
- matrix.cc
- matrix.h
- point.cc
- point.h
- pointf.cc
- pointf.h
- polarMetrics.h
- utils.cc
- utils.h

CaptureLib
- GrayMap.h
- GrayMapXIL.cc
- GrayMapXIL.h
- GXILGrabber.cc
- GXILGrabber.h
- GXILImage.cc
- GXILImage.h
- GXILWindow.cc
- GXILWindow.h

WindowLib
- FCalibrationWindow.cc
- FCalibrationWindow.h
- FHistogramWindow.cc
- FHistogramWindow.h
- FImageWindow.cc
- FImageWindow.h

WindowLibMac
- FWindow.cc
- FWindow.h

WindowLibX
- FDisplay.cc
- FDisplay.h
- FStereoWindow.cc
- FStereoWindow.h
- FWindow.cc
- FWindow.h

The rest of this appendix lists source code from a selected number of files.

Two-Dimensional Points

```
template<class N> struct Point2
{
   typedef N ElemType;

   N x, y;

   Point2( void ){}
   Point2( N nx, N ny ) : x(nx), y(ny) {}

   Point2<N> operator-( void ) const { return Point2<N>(-x,-y); }

   Point2<N> &operator*=( N r ){ x*=r; y*=r; return *this; }
   Point2<N> &operator/=( N r ){ x/=r; y/=r; return *this; }

   Point2<N> &operator+=( const Point2<N> &r){ x+=r.x; y+=r.y; return *this; }
   Point2<N> &operator-=( const Point2<N> &r){ x-=r.x; y-=r.y; return *this; }
   Point2<N> &operator*=( const Point2<N> &r){ x*=r.x; y*=r.y; return *this; }
   Point2<N> &operator/=( const Point2<N> &r){ x/=r.x; y/=r.y; return *this; }

   N dot  ( const Point2<N> &r ) const { return x*r.x + y*r.y; }
   N cross( const Point2<N> &r ) const { return x*r.y - y*r.x; }
};


template<class N> inline
Point2<N> operator*( N l, const Point2<N> &r ){ return Point2<N>(r) *= l; }

template<class N> inline
Point2<N> operator/( N l, const Point2<N> &r ){ return Point2<N>(l/r.x,l/r.y); }

template<class N> inline
Point2<N> operator*( const Point2<N> &l, N r ){ return Point2<N>(l) *= r; }

template<class N> inline
Point2<N> operator/( const Point2<N> &l, N r ){ return Point2<N>(l) /= r; }


template<class N> inline
Point2<N> operator+( const Point2<N> &l, const Point2<N> &r ){return Point2<N>(l)+=r;}

template<class N> inline
Point2<N> operator-( const Point2<N> &l, const Point2<N> &r ){return Point2<N>(l)-=r;}

template<class N> inline
Point2<N> operator*( const Point2<N> &l, const Point2<N> &r ){return Point2<N>(l)*=r;}

template<class N> inline
Point2<N> operator/( const Point2<N> &l, const Point2<N> &r ){return Point2<N>(l)/=r;}


template<class N> inline N abs2( const Point2<N> &v ){ return v.dot(v); }
template<class N> inline N abs(  const Point2<N> &v ){ return sqrt( v.dot(v) ); }
template<class N> inline N arg(  const Point2<N> &v ){ return atan2( v.y, v.x ); }


template<class N> inline
Point2<N> rtop( const Point2<N> &p ){ return Point2<N>(abs(p),arg(p)); }

template<class N> inline
Point2<N> ptor( const Point2<N> &p ){ return Point2<N>(p.x*cos(p.y),p.x*sin(p.y)); }


typedef Point2<float> PointF2;
```

Three-Dimensional Points

```
template<class N> struct Point3
{
   typedef N ElemType;

   N x, y, z;

   Point3( void ){}
   Point3( N nx, N ny, N nz ) : x(nx), y(ny), z(nz) {}

   Point3<N> operator-( void ) const { return Point3<N>(-x,-y,-z); }

   Point3<N> &operator*=( N r ){ x*=r; y*=r; z*=r; return *this; }
   Point3<N> &operator/=( N r ){ x/=r; y/=r; z/=r; return *this; }

   Point3<N> &operator+=( const Point3<N> &r ){x+=r.x; y+=r.y; z+=r.z; return *this;}
   Point3<N> &operator-=( const Point3<N> &r ){x-=r.x; y-=r.y; z-=r.z; return *this;}
   Point3<N> &operator*=( const Point3<N> &r ){x*=r.x; y*=r.y; z*=r.z; return *this;}
   Point3<N> &operator/=( const Point3<N> &r ){x/=r.x; y/=r.y; z/=r.z; return *this;}

   N dot( const Point3<N> &r ) const { return x*r.x + y*r.y + z*r.z; }
   Point3<N> cross( const Point3<N> &r ) const
   {   return Point3<N>( y*r.z - z*r.y, z*r.x - x*r.z, x*r.y - y*r.x ); }
};


template<class N> inline
Point3<N> operator*( N l, const Point3<N> &r ){ return Point3<N>(r) *= l; }

template<class N> inline
Point3<N> operator/( N l, const Point3<N> &r ){ return Point3<N>(l/r.x,l/r.y,l/r.z); }

template<class N> inline
Point3<N> operator*( const Point3<N> &l, N r ){ return Point3<N>(l) *= r; }

template<class N> inline
Point3<N> operator/( const Point3<N> &l, N r ){ return Point3<N>(l) /= r; }


template<class N> inline
Point3<N> operator+( const Point3<N> &l, const Point3<N> &r ){return Point3<N>(l)+=r;}

template<class N> inline
Point3<N> operator-( const Point3<N> &l, const Point3<N> &r ){return Point3<N>(l)-=r;}

template<class N> inline
Point3<N> operator*( const Point3<N> &l, const Point3<N> &r ){return Point3<N>(l)*=r;}

template<class N> inline
Point3<N> operator/( const Point3<N> &l, const Point3<N> &r ){return Point3<N>(l)/=r;}


template<class N> inline N abs2( const Point3<N> &v ){ return v.dot(v); }
template<class N> inline N abs ( const Point3<N> &v ){ return sqrt( v.dot(v) ); }


// ****  Nifty utilities.  ****
template<class PointType> inline
PointType normalize( const PointType &v ){ return v/abs(v); }

template<class N> inline
Point3<N> homogenize( const Point2<N> &p, float w=1 ){ return Point3<N>(p.x,p.y,w); }

template<class N> inline
Point2<N> project( const Point3<N> &P ){ return Point2<N>(P.x,P.y)/P.z; }


typedef Point3<float> PointF3;
```

Lines

```
// ****  Two dimensional lines.  ****
inline PointF3 line( const PointF2 &p0, const PointF2 &p1 )
{
   return PointF3( p1.y - p0.y, p0.x - p1.x, p1.x*p0.y - p0.x*p1.y );
}


inline PointF3 normalizeLine( const PointF3 &l )
{
   return l/(float)sqrt(l.x*l.x + l.y*l.y);
}


inline PointF2 lineToPolar( const PointF3 &l )
{
   return PointF2( -l.z/sqrt(l.x*l.x + l.y*l.y), atan2(l.y,l.x) );
}


inline PointF3 polarToLine( const PointF2 &l )
{
   return PointF3( cos(l.y), sin(l.y), -l.x );
}


inline PointF2 IntersectLines( const PointF3 &l1, const PointF3 &l2 )
{
   return project( l1.cross(l2) ); // Cramer's Rule
}


// ****  Three dimensional lines.  ****
PointF3 IntersectLines( const PointF3 &dP, const PointF3 &P0,
                        const PointF3 &dQ, const PointF3 &Q0 )
{
   //  Equations (3.4)
   PointF3 l1( -dP.dot(dP), dP.dot(dQ), dP.dot(Q0 - P0) );
   PointF3 l2( -dP.dot(dQ), dQ.dot(dQ), dQ.dot(Q0 - P0) );

   PointF2 st = IntersectLines( l1, l2 );

   // Equation (3.5)
   return (st.x*dP + P0 + st.y*dQ + Q0)/2.0f;
}
```

Three-D Rotations

```
matrix rotX( float omega )
{
   matrix R(3,3,matrix::eye);

   // Equation (2.3a)
   R(2,2) = cos(omega); R(2,3) = sin(omega);
   R(3,2) = -R(2,3);    R(3,3) = R(2,2);

   return R;
}


matrix rotY( float phi )
{
   matrix R(3,3,matrix::eye);

   // Equation (2.3b)
   R(1,1) = cos(phi); R(1,3) = -sin(phi);
   R(3,1) = -R(1,3);  R(3,3) = R(1,1);

   return R;
}


matrix rotZ( float kappa )
{
   matrix R(3,3,matrix::eye);

   // Equation (2.3c)
   R(1,1) = cos(kappa); R(1,2) = sin(kappa);
   R(2,1) = -R(1,2);    R(2,2) = R(1,1);

   return R;
}


matrix rotate( const PointF3 &angles )
{
   // Equation (2.2)
   return rotZ(angles.z)*rotY(angles.y)*rotX(angles.x);
}


PointF3 rotationToAngles( const matrix &R )
{
   // Equations (2.5)
   return PointF3( atan2(-R(3,2),R(3,3)), asin(R(3,1)), atan2(-R(2,1),R(1,1)) );
}
```

Camera Model

```
struct Camera
{
   matrix  R;
   PointF3 T;
   PointF2 f;
   PointF2 o;
   float   k;

   Camera( void ) : R(3,3,matrix::eye), T(0,0,0), f(1,1), o(0,0), k(0) {}

   Camera( const matrix &rotation, const PointF3 &translation,
      const PointF2 &focalLength, const PointF2 &origin, float distortion )
      : R(rotation), T(translation), f(focalLength), o(origin), k(distortion)
   {
      if( R.nrows()!=3 || R.ncols()!=3 ) throw matrix::BadSize();
   }

   matrix getParameters( void ) const;
   void   setParameters( const matrix &m );

   PointF2 worldToImage ( const PointF3 &P ) const
   { return cameraToImage( worldToCamera(P) ); }
   PointF3 worldToCamera( const PointF3 &P ) const { return R*P+T; } // Equation (2.1)
   PointF2 cameraToImage( const PointF3 &P ) const
   { return projectionToImage( project(P) ); }
   PointF2 projectionToImage( const PointF2 &pq ) const;

   PointF2 imageToProjection( const PointF2 &p ) const;
   PointF3 cameraToWorld( const PointF3 &P ) const { return transpose(R)*(P-T); }
};


matrix Camera::getParameters( void ) const
{
   matrix m( 11, 1 );

   int i = 1;
   m(i++,1) = T.x; m(i++,1) = T.y; m(i++,1) = T.z;
   PointF3 angles = rotationToAngles( R );
   m(i++,1) = angles.x; m(i++,1) = angles.y; m(i++,1) = angles.z;
   m(i++,1) = o.x; m(i++,1) = o.y;
   m(i++,1) = f.x; m(i++,1) = f.y;
   m(i++,1) = k;

   return m;
}


void Camera::setParameters( const matrix &m )
{
   if( m.nrows()!=11 || m.ncols()!=1 ) return; // Uh, oh!

   int i = 1;
   T.x = m(i++,1); T.y = m(i++,1); T.z = m(i++,1);
   R = rotate(PointF3(m(i,1),m(i+1,1),m(i+2,1))); i+=3;
   o.x = m(i++,1); o.y = m(i++,1);
   f.x = m(i++,1); f.y = m(i++,1);
   k = m(i++,1);
}


PointF2 Camera::imageToProjection( const PointF2 &xy ) const
{
   // Equation (2.7)
   PointF2 uv = (xy - o)/f;

   // Equation (2.6)
   return (1 + k*(uv.x*uv.x + uv.y*uv.y))*uv;
}
```

Camera Model (Contd.)

```
PointF2 Camera::projectionToImage( const PointF2 &pqs ) const
{
   PointF2 uv       = pqs;
   PointF2 minUV    = uv;
   float   minError = FLT_MAX;

   while( true )
   {
      PointF2 uv2 = uv*uv;
      float   d   = 1 + k*(uv2.x + uv2.y);
      PointF2 eps = pqs - d*uv; // Equation (2.8)

      // ****  Calculate error (before updating solution).  ****
      float error = eps.dot(eps);
      if( error>=minError ) break;
      minError = error; minUV = uv;
      if( error==0 ) break;

      // ****  Calculate new solution.  Equation (2.9)  ****
      PointF3 J1( d + 2*k*uv2.x, 2*k*uv.x*uv.y, -eps.x );
      PointF3 J2( 2*k*uv.x*uv.y, d + 2*k*uv2.y, -eps.y );

      uv += IntersectLines( J1, J2 ); // Cramer's Method
   }

   // Equation (2.7)
   return minUV*f + o;
}
```

Linear Camera Calibration

```
void linearCalibration( const PointF2 p[], const PointF3 P[], int N, Camera &C )
{
   DLT( p, P, N, C );
   extractCameraParams( C );
}


void DLT( const PointF2 p[], const PointF3 P[], int N, Camera &C )
{
   matrix A(2*N,11), b(2*N,1);
   for( int i=0; i<N; i++ ) rowDLT( A, b, p[i], P[i], 2*i+1 );

   // ****  Find Solution Using SVD.  ****
   matrix w = pinv(A)*b;

   C.R(1,1) = -w(1,1); C.R(1,2) = -w( 2,1); C.R(1,3) = -w( 3,1); C.T.x = -w(4,1);
   C.R(2,1) = -w(5,1); C.R(2,2) = -w( 6,1); C.R(2,3) = -w( 7,1); C.T.y = -w(8,1);
   C.R(3,1) =  w(9,1); C.R(3,2) =  w(10,1); C.R(3,3) =  w(11,1); C.T.z =  1;
}


void rowDLT( matrix &A, matrix &b, const PointF2 &p, const PointF3 &P, int i )
{
   // Equation (2.14)

   A(i  ,1)=P.x; A(i  ,2)=P.y; A(i  ,3)=P.z; A(i  ,4)=1;
   A(i+1,1)=  0; A(i+1,2)=  0; A(i+1,3)=  0; A(i+1,4)=0;

   A(i  ,5)=  0; A(i  ,6)=  0; A(i  ,7)=  0; A(i  ,8)=0;
   A(i+1,5)=P.x; A(i+1,6)=P.y; A(i+1,7)=P.z; A(i+1,8)=1;

   A(i  ,9)=p.x*P.x; A(i  ,10)=p.x*P.y; A(i  ,11)=p.x*P.z;
   A(i+1,9)=p.y*P.x; A(i+1,10)=p.y*P.y; A(i+1,11)=p.y*P.z;

   b(i  ,1)=-p.x;
   b(i+1,1)=-p.y;
}


void extractCameraParams( Camera &C )
{
   // ****  Normalize.  Equation (2.16)  ****
   if( C.T.z<0 ){ C.T = -C.T; C.R = -C.R; }  // Assume target is in front of camera.

   PointF3 R1, R2, R3;
   GetRow(C.R,R3,3);

   C.T /= abs(R3); C.R /= abs(R3);
   GetRow(C.R,R1,1); GetRow(C.R,R2,2); GetRow(C.R,R3,3);

   // ****  Principal point.  Equations (2.17a) and (2.17b)  ****
   C.o.x = R1.dot(R3); C.o.y = R2.dot(R3);

   // ****  Focal point/Scaling factor.  Equations (2.17c) and (2.17d)  ****
   R1 -= C.o.x*R3, R2 -= C.o.y*R3;

   C.f.x = abs(R1); C.f.y = abs(R2);

   // ****  Translation vector.  Equation (2.17e)  ****
   C.T.x = (C.T.x - C.o.x*C.T.z)/C.f.x;
   C.T.y = (C.T.y - C.o.y*C.T.z)/C.f.y;

   // ****  Rotation matrix.  Equation (2.17f)  ****
   R1 /= C.f.x; R2 /= C.f.y;

   PutRow(C.R,R1,1); PutRow(C.R,R2,2); PutRow(C.R,R3,3);
}
```

Nonlinear Camera Calibration

```
struct CalInfo
{
   const PointF2 *p;
   const PointF3 *P;
   int            N;
   Camera        *C;

   CalInfo( const PointF2 *points, const PointF3 *Points, int numPoints,
      Camera *cam ) : p(points), P(Points), N(numPoints), C(cam) {}
};


float ncError( const matrix &x, void *info )
{
   const PointF2 *p = ((CalInfo *)info)->p;
   const PointF3 *P = ((CalInfo *)info)->P;
   Camera        *C = ((CalInfo *)info)->C;
   int            N = ((CalInfo *)info)->N;

   C->setParameters( x );

   // Equation (2.18)
   float sum = 0;
   for( int i=0; i<N; i++ )
      sum += abs2( project(C->worldToCamera(P[i])) - C->imageToProjection(p[i]) );

   return sum;
}


void nonlinearCalibration( const PointF2 p[], const PointF3 P[], int N, Camera &C)
{
   // ****  Initial condition.  ****
   matrix x = C.getParameters();

   float prevError = FLT_MAX;
   CalInfo info( p, P, N, &C );
   for( int i=0; i<200; i++ )
   {
      long evals = 5000;
      float error = simplex( x, ncError, &info, 1e-3, 1e-3, evals );

      // ****  Run the simplex optimizer until the error stops changing.  ****
      if( fabs(prevError - error) < 0.000001*error ) break;
      prevError = error;
   }

   // ****  The answer  ****
   C.setParameters( x );
}
```

Triangulation

```
PointF3 Triangulate( const PointF2 &pp[2], const Camera C[2] )
{
    PointF3 d0 = C[0].R*homogenize( C[0].imageToProjection(pp[0]) );
    PointF3 d1 = C[1].R*homogenize( C[1].imageToProjection(pp[1]) );

    return IntersectLines( d0, C[0].T, d1, C[1].T );
}
```

Fitting a Line to an Image Edge

```
float minGrad( const matrix &x, void *info )
{
   const SignedImage *I = (SignedImage *)info;

   PointF3 l = polarToLine( PointF2(x(1,1),x(2,1)) );

   int w = I->width(), h = I->height();

   // Equation (3.7)
   float sum = 0;
   if( fabs(l.y) > fabs(l.x) )
      for( int x=0; x<w; x++ )
      {
         int y = -(l.z + x*l.x)/l.y + 0.5;

         if( y>=0 && y<h ) sum += (*I)(x,y);
      }
   else
      for( int y=0; y<h; y++ )
      {
         int x = -(l.z + y*l.y)/l.x + 0.5;

         if( x>=0 && x<w ) sum += (*I)(x,y);
      }

   // ****  Return negative to find maximum.  ****
   return -sum;
}


PointF3 FindEdge( const SignedImage &I, const PointF2 &p0, const PointF2 &p1, int b )
{
   // ****  Initial condition.  ****
   PointF2 polar = lineToPolar( line(p1,p0) );

   matrix x(2,1);
   x(1,1) = polar.x; // r
   x(2,1) = polar.y; // theta

   float prevError = FLT_MAX;
   for( int i=0; i<20; i++ )
   {
      long evals = 5000;
      float error = simplex( x, minGrad, &I, 1e-5, 1e-5, evals );

      // **** Run the simplex optimizer until the error stops changing. ****
      if( error>=prevError ) break;
      prevError = error;
   }

   // ****  The answer  ****
   return polarToLine( PointF2(x(1,1),x(2,1)) );
}
```

Polar Metrics

```
inline float reflect( float angle ){ angle += angle<0 ? pi : -pi; return angle; }


inline float angleDiff( float a1, float a2 )
{
   float angle = fabs(a1 - a2);
   return angle>pi ? 2*pi - angle : angle;
}


inline int angleEq( double a1, double a2, double tol ){ return angleDiff(a1,a2)<tol; }


struct Differences
{
   PointF2 *diff;
   long     size;
   long     windowSize;

   Differences( const PointF2 in[], long itsSize, long itsWindowSize )
   {
      size       = itsSize;
      windowSize = itsWindowSize;
      diff       = new PointF2[windowSize*(size-1)];

      // ****  Calculate polar distances for each point.  ****
      for( long j=0, k=0; j<size-1; j++ )
         for( long i=1; i<=windowSize; i++ )
            diff[k++] = i+j<size ? rtop( in[i+j] - in[j] ) : PointF2( 0, 0 );
   }

   ~Differences( void ){ delete[] diff; }

         PointF2 &operator[]( int i )       { return diff[i]; }
   const PointF2 &operator[]( int i ) const { return diff[i]; }
};


template<class SearchFcn>
void searchForPoint( long start, const Differences &diff, SearchFcn &funct )
{
   const long size       = diff.size;
   const long windowSize = diff.windowSize;
   const long di         = windowSize - 1;

   // ****  Search diagonal.  ****
   long k   = max( start - windowSize, 0L );
   long beg = k*di + start - 1;
   long mid = start*windowSize - 1;
   for( long i=beg; i<mid; i+=di, k++ ) funct( diff[i].x, reflect(diff[i].y), k );

   // ****  Search across.  ****
   long end = min( size - start, windowSize ) + i;
   for( i++, k++; i<end; i++, k++ ) funct( diff[i].x, diff[i].y, k );
}


struct ClosestForDirection
{
   long    minPos;
   PointF2 minDist;
   float   dir, tol;

   ClosestForDirection( float direction, float tolerance )
   { dir = direction; tol = tolerance; minPos = -1; minDist.x = FLT_MAX; }

   void operator()( float dist, float angle, long index )
   {
      if( angleEq(angle,dir,tol) && dist<minDist.x )
      { minPos = index; minDist = PointF2(dist,angle); }
   }
};
```

Sorting a Grid of Points into Rows and Columns

```
long findPoint( long start, const Differences &diff, double forwardDir )
{
   ClosestForDirection k( forwardDir, pi/8 );
   searchForPoint( start, diff, k );

   return k.minPos;
}


inline long findLeft( long start, const Differences &diff )
{
   return findPoint( start, diff, pi );
}


inline long findRight( long start, const Differences &diff )
{
   return findPoint( start, diff, 0 );
}


inline long findTop( long start, const Differences &diff )
{
   return findPoint( start, diff, pi/2 );
}


inline long findBottom( long start, const Differences &diff )
{
   return findPoint( start, diff, -pi/2 );
}


long findTopLeft( const Differences &diff )
{
   long topLeft = -1;
   for( long n=1       ; n>=0; n=findLeft(n,diff) ) topLeft = n;
   for( long n=topLeft; n>=0; n=findTop(n,diff)  ) topLeft = n;
   return topLeft;
}


PointF2 SortGrid( const PointF2 in[], PointF2 out[], long size )
{
   const long WindowSize = 30;

   // ****  Calculate polar differences.  ****
   Differences diff( in, size, WindowSize );

   // ****  Sort points.  ****
   long cols  = 0;
   long count = 0;
   int  isFirstRow = 1;
   for( long m=findTopLeft(diff); m>=0; m=findTop(m,diff) )
   {
      for( long n=m; n>=0; n=findRight(n,diff) ) out[count++] = in[n];

      if( isFirstRow ){ cols = count - 1; isFirstRow = 0; }
   }

   return PointF2( size/cols, cols );
}
```

Vita

Geoffrey Yerem was born in Glendale, New York on November 9, 1968. He graduated from Roslyn High School in Roslyn, New York. In May 1996, he obtained a Bachelor of Science degree in Electrical Engineering at the University of Tennessee, Knoxville where he studied analog signal processing, electronics, image processing, and robotics. In December 2001, he obtained his Master of Science degree in Electrical Engineering at UT, Knoxville specializing in signal processing and computer vision.