Real Time Fusion of Radioisotope Direction Estimation and Visual Object Tracking

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Elliot Davis Greenlee

December 2018

© by Elliot Davis Greenlee, 2018
All Rights Reserved.

I dedicate my research to my family, Angela McCabe, and Jennifer McComas, without whom I would not have made it.

Acknowledgments

First, I would like to thank my advisor, Dr. Hairong Qi, for providing me with the opportunity to do research in these fields for the last two years. Without her I would have been lost from start to finish. The custom blend of patience, freedom, and guidance she has afforded me has facilitated my growth and discovery throughout my graduate academic career.

Next, I would also like to thank my committee members Dr. Mark Dean and Dr. Jason Hayward for providing leadership, advice, and guidance in the form of constructive suggestions both in the scope of my research and about my future plans.

My absolute gratitude goes to The University of Tennessee and the Electrical Engineering and Computer Science Department. By preparing me for graduate school, and consistently supporting me financially through scholarships and graduate assistantships, they laid the groundwork upon which all my effort has been possible.

My colleagues across my research groups were always a source of inspiration, advice, and competition to do better. I am especially thankful to my office mates, who patiently tried to answer all of my questions, and to my companions who worked with me in various competitions.

Lastly, I owe back to my parents, friends, and girlfriend Angela the sum total of all the support and encouragement they have given me over the last two years. Thanks to Tammy for asking me every day why I hadn't started working yet. Without their duties as a talking sounding board, my work would still be in my head. Finally, thank you for keeping me sane by reminding me about the important things throughout this process.

Abstract

Research into discovering prohibited nuclear material plays an integral role in providing security from terrorism. Although many diverse methods contribute to defense, there exists a capability gap in localizing moving sources. This thesis introduces a real time radioisotope tracking algorithm assisted by visual object tracking methods to fill the capability gap. The proposed algorithm can estimate carrier likelihood for objects in its field of view, and is designed to assist a pedestrian agent wearing a backpack detector. The complex, crowd-filled, urban environments where this algorithm must function combined with the size and weight limitations of a pedestrian system makes designing a functioning algorithm challenging.

The contribution of this thesis is threefold. First, a generalized directional estimator is proposed. Second, two state-of-the-art visual object detection and visual object tracking methods are combined into a single tracking algorithm. Third, those outputs are fused to produce a real time radioisotope tracking algorithm. This algorithm is designed for use with the backpack detector built by the IDEAS for WIND research group. This setup takes advantage of recent advances in detector, camera, and computer technologies to meet the challenging physical limitations.

The directional estimator operates via gradient boosting regression to predict radioisotope direction with a variance of 50 degrees when trained on a simple laboratory dataset. Under conditions similar to other state-of-the-art methods, the accuracy is comparable. YOLOv3 and SiamFC are chosen by evaluating advanced visual tracking methods in terms of speed and efficiency across multiple architectures, and in terms of accuracy on datasets like the Visual Object Tracking (VOT) Challenge and Common Objects in Context (COCO). The resultant tracking algorithm operates in real time. The outputs of direction estimation and visual tracking are fused using sequential Bayesian inference to predict carrier likelihood.

Using lab trials evaluated by hand on visual and nuclear data, and a synthesized challenge dataset using visual data from the Boston Marathon attack, it can be observed that this prototype system advances the state-of-the-art towards localization of a moving source.

Table of Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Background . . . 2

1.2.1 Nuclear Detection . . . 2

1.2.2 Machine Learning . . . 6

1.2.3 Visual Algorithms . . . 17

1.3 Related Works . . . 21

1.3.1 Nuclear Localization . . . 21

1.3.2 Object Detection . . . 23

1.3.3 Object Tracking . . . 25

1.4 Contributions . . . 29

1.5 Thesis Outline . . . 30

2 Nuclear Directionality **31**

2.1 Algorithm Design . . . 31

2.2 Results . . . 33

2.2.1 Simulated and Real Detector Setups . . . 33

2.2.2 Simulation Data . . . 36

2.2.3 Lab Data . . . 36

2.2.4 Algorithm Design . . . 36

3 Visual Object Tracking **41**

3.1 Object Detection . . . 41

3.1.1 Datasets . . . 42
3.1.2 Metrics . . . 42
3.1.3 Choosing YOLOv3 . . . 43
3.2 Object Tracking . . . 48
3.2.1 Datasets . . . 48
3.2.2 Metrics . . . 49
3.2.3 Choosing SiamFC . . . 50
3.3 Results . . . 54
3.3.1 Data . . . 54
3.3.2 Object Detection . . . 55
3.3.3 Object Tracking Results . . . 59
3.3.4 Multi-Object Tracking . . . 59

4 Active Tracking System **63**
4.1 System Overview . . . 63
4.1.1 Nuclear Directionality . . . 65
4.1.2 Visual Object Tracking . . . 65
4.2 Sequential Bayesian Inference . . . 66
4.3 Results . . . 67

5 Conclusions and Future Work **71**

Bibliography **73**

Vita **86**

List of Tables

2.1 The synthetic and laboratory settings explore different experimental variables. 33

2.2 This table shows the resulting error for each standard regression algorithm using the simulated data. 38

2.3 Training on an 8.15 μCi Cs-137 source and testing on an 18.06 μCi Cs-137 source at 0.91 meters using three different ratio preprocessing techniques shows that ratios are necessary for the algorithm to generalize to different relative source strengths. Counts are recorded for four seconds, and random forest regression is used for the predictions. 39

2.4 As the counting time increases, the error decreases. These are the results at 0.91 meters using an 18.06 μCi Cs-137 source. Random forest regression is used for prediction given individual ratio features. 39

2.5 Five experimental data setups are used to evaluate various regression methods (A: Cs-137/8.15 μCi/0.53m, B: Cs-137/8.15 μCi/0.91m, C: Cs-137/9.91 μCi/0.91m, D: Cs-137/18.06 μCi/0.91m, and E: Co-60/0.31 μCi/0.91m). The lowest three rmse values for each setup are highlighted in red. 39

2.6 The error increases when changing the source material between training and testing. Each setup is recorded at 0.91 meters. The error for Cobalt 60 is much higher because the source strength of the Cobalt source is 0.31 μCi, while for Cesium it is 8.15 μCi. It appears that using ratios also reduces the error when switching source materials, but this is likely due to compensating for the large difference in source strengths of these two experimental setups. 40

2.7 Although the error increases when changing the distance between training and testing, using ratios reduces the difference in error. Each setup is recorded at 0.815 μCi. 40
2.8 Although the error increases when changing the source strength between training and testing, using ratios reduces the difference in error. Each setup is recorded at 0.91 meters. 40

3.1 Older YOLO methods achieve state-of-the-art accuracy on the VOC 2007+2012 datasets while maintaining efficiency. The fastest speed and highest mAP for an IoU of 50 are highlighted in red. For networks with implementations operating at different resolutions, multiple results are provided with the input resolution given. 43
3.2 YOLOv3 achieves state-of-the-art accuracy on the COCO dataset while maintaining efficiency. The fastest speed and highest mAP for an IoU of 50 are highlighted in red. For networks with implementations operating at different resolutions, multiple results are provided with the input resolution given. 44
3.3 SiamFC is the winner of the VOT real time 2017 challenge. The results shown are the top five entries to the real time portion only. The highest EAO is highlighted in red. 52
3.4 SiamFC achieves state-of-the-art performance on the standard VOT 2017 challenge. Various methods from the top five 2017 methods, the real time competition, and dual 2016-2017 competition entries are shown. The highest EAO is highlighted in red. 52
3.5 SiamFC achieves state-of-the-art performance as a balance between EAO and speed. These entries all competed in some form in both the 2016 and 2017 VOT challenges. All results below are obtained on the 2016 dataset. 52
3.6 The difference in fps between GPU and CPU architectures for YOLOv3 is large. The fps values from running on each Boston bombing sequence are shown. 55

3.7 SiamFC tracks at 10 times the frames per second on a GPU compared to a CPU. 59

4.1 This table shows two iterations of Bayesian updating for three objects. Starting with equal priors, the angle relative to the source is given for each object at each step, and the corresponding posteriors are calculated as above using Bayes' theorem. 67

4.2 The fusion algorithm takes multiple iterations to reach 80% confidence when five possible carriers are present at a given angular split from each other. . . 67

List of Figures

1.1 Different types of radiation are halted by different shields [1]. 3
1.2 Interactions in the detector are recorded as pulses. (Knoll fig 4.1, 4.2) 5
1.3 A simplified photomultiplier (Knoll fig 9.1) 7
1.4 An example dataset showing various components. 8
1.5 A simple unsupervised clustering example from Analytics Vidhya [2]. 9
1.6 A simple 2D separating hyperplane [3]. 12
1.7 A simple decision tree to predict car prices. 13
1.8 An example artificial neuron [4]. 15
1.9 An example fully-connected artificial neural network with a single hidden layer [5]. 16
1.10 Two handwritten letters and their Fourier transformations from Michelle Dunn at Swinburne University of Technology [6]. 19
1.11 The LeNet5 architecture [7]. 20
1.12 Trackers attempt to place an estimated bounding box A_T with as large an overlap (yellow) as possible with the ground truth A_G. (Background image from [8]) 26
1.13 Papers submitted indicates fraction of total papers. Over the past five years, CNN and correlation filter methods have become dominant [8]. 27

2.1 This is an example of simulated data when recorded in the three detector arrangement. This image was produced by Carl Britt. 34
2.2 The detector setup and experimental apparatus used to record the training and evaluation dataset. 35

2.3 Counts over one second intervals for the 8.15 μCi cesium-137 source at 0.53 meters. 36
2.4 The red line is the Gaussian calculated using the mean and variance of all prediction data from gradient boosting. 38

3.1 In terms of speed, YOLOv3 is a class ahead. The 'Other' methods listed are taken from Table 3.2. The trends in speed and runtime for YOLO and RetinaNet are caused by varying network input size. 45
3.2 The backbone network for YOLOv3 is darknet-53, a classification network (from Redmon, 2018 [9]). 47
3.3 YOLOv3 predicts x and y offset from the location of filter application and w and h scaling from priors of set size. 48
3.4 In order to calculate equivalent filter operations, a baseline for the hardware is calculated by applying a max operation on each 30x30 window of a 600x600 image (from [8]). 50
3.5 SiamFC strikes a balance between accuracy and speed. These are the VOT 2016 Challenge results from Table 3.5. 51
3.6 The backbone architecture of the of the convolutional function is similar to AlexNet (from Bertinetto, 2016 [10]). 53
3.7 A score map is output for all translated sub-windows in the search image (from Bertinetto, 2016 [10]). 54
3.8 Running the YOLOv3 tiny and normal networks on Boston sequence 1 results in a difference in detection capabilities. (tiny a-d, normal e-h) 56
3.9 In crowds, the tiny network misses many partially occluded or rotated individuals. (tiny a-d, normal e-h) 56
3.10 One artifact of the network at a low confidence threshold is the tendency to mislabel the entire crowd as one person. (tiny a-d, normal e-h) 57
3.11 The normal network can capture many individuals in a crowd, but is overwhelmed by crowds appearing farther back in frame. (tiny a-d, normal e-h) 57

3.12 Sequences like this, with a single individual, give the normal network no challenge. (tiny a-d, normal e-h) . . . 58
3.13 On sequence six, the tiny network has trouble detecting the target, and misses the majority of the parked cars (tiny a-d, normal e-h) . . . 58
3.14 Both subjects are successfully tracked. . . . 60
3.15 One of the subjects is lost due to occlusion. . . . 60
3.16 Here, the backpack rotation and crowd occlusion cause a loss of one subject. 60
3.17 Provided with a clear, unoccluded face, the network can achieve high accuracy with little information. . . . 61
3.18 Even though the subject rotates, tracking continues. . . . 61
3.19 Despite the low resolution and partial occlusion, SiamFC tracks the subject into the car. . . . 61
3.20 Both targets are detected and then tracked for the duration of the sequence. 61
3.21 Both targets are detected, but only one is tracked for the duration of the sequence. . . . 62
3.22 As the targets walks through the crowd, he comes into view, is detected, and then is tracked until his body is occluded again. . . . 62
3.23 Even when initialized by the detector, the tracker can still remain robust to the target's rotation. . . . 62
3.24 Although the target is lost to the detector over the course of the sequence as seen above, the tracker is able to continue to localize given the detector's initial bounding box. . . . 62

4.1 An overview of the system flow path. . . . 64
4.2 A Gaussian distribution is used to calculate likelihood for the planets, OK Go, and Beyonce sequences. . . . 69
4.3 Four frames from a MATLAB simulation of orbiting bodies around a detector. The red body has the source, and over time the system predictions increase in accuracy. Video created by Steven Patrick. . . . 69

4.4 Four frames from Ok Go's music video "White Knuckles" [11] are shown. Dale, outlined in blue, is holding a fabricated source at his center of mass. The tracking information is input manually. Video created by Steven Patrick. 69
4.5 Four frames from Beyonce's music video "All the Single Ladies" [12] are shown. Beyonce is holding a fabricated source at her center of mass. The tracking information is input manually. Video created by Steven Patrick. 70
4.6 Four frames from the Boston marathon bombing video dataset are shown. The suspect outlined in a white circle is holding a fabricated source at his center of mass. The tracking information is calculated using YOLOv3 and SiamFC. 70

Chapter 1

Introduction

1.1 Motivation

Since 1970, with the signing of the Treaty on the Non-Proliferation of Nuclear Weapons, the world has committed to controlling usage of nuclear resources, and has aspired to disarmament. In the United States of America, the Department of Homeland Security (DHS) acts to protect citizens from threats posed by special nuclear material. The consequences of terrorism ensure that efforts must always be made towards improving the competence of defensive and surveillance systems.

Fortunately, new technological advances have paved the way for various improvements. Nuclear detectors can be made smaller, lighter, and more sensitive, computers shrink while their computational power increases, and new algorithms achieve superior accuracies with accelerated runtimes. These developments allow smaller deployments using drone or pedestrian agents. Current commercial pedestrian systems do not incorporate recent state-of-the-art improvements, leaving a capability gap open.

In order for a proposed system to succeed, it must meet a suite of challenges in functionality. The extreme size and weight limitations of pedestrian systems propagate into constraints to the detector capability, computational power, and number of sensors. Such a system would also need to function in typically urban regions with complex topography and dynamic crowds, possibly remaining inconspicuous in such a cluttered environment. Finally,

real time feedback is required; as the sources and agents move, immediate assistance towards the goals of detection, identification, and localization should be provided.

Research to address these challenges is being performed as part of the five year Investigation of DEtectors, Algorithms, and Systems for Wearable Intelligent Nuclear Detection (IDEAS for WIND) project for the Department of Homeland Security (DHS). Researchers for this project come from the University of Tennessee's Department of Electrical Engineering and Computer Science and Department of Nuclear Engineering, Oak Ridge National Laboratory (ORNL), and Pacific Northwest National Laboratory (PNNL). The final goal of the five year project is to create a wearable nuclear detection system with optimized size, weight, and power that has gamma/neutron detection response, modularity, spatial tracking, localization, networking and reachback capability, and smart device integration. Along the way, research and projects related to these and a myriad of other subjects are being completed by the team.

The rest of Chapter 1 provides readers of various backgrounds with the information necessary to both understand and contextualize this thesis.

1.2 Background

Since this thesis spans both nuclear and computer vision fields, different readers may need a brief introduction to the basics of those fields. The following sections introduce nuclear detection, machine learning, and visual algorithms in a way that relates to the research explained in this thesis. Familiar readers should start with section 1.3.

1.2.1 Nuclear Detection

This section provides a brief overview to those who are unfamiliar with radioisotopes, radiation, and detectors. First, types of radiation are discussed, and then detector mechanics are explained.

Energy is emitted from matter as both rays and particles. This radiation can be non-ionizing, such as heat or visible light, or ionizing, such has x-rays or particles. Non-ionizing radiation travels through materials and delivers energy, while higher energy ionizing radiation

breaks molecular bonds. Atoms with a varying amount of neutrons, yet the same number of protons, are referred to as isotopes of an element. Typical sources are unstable radioisotopes with a high degree of nuclear energy which is emitted as ionizing radiation [13].

There are various classifications of observable sources. Nuisance sources include naturally occurring radioactive materials (NORM) which can be terrestrial, cosmic, or even the detector itself, and legitimate sources such as for medical uses like gamma imaging. Target materials include fissile isotopes and special nuclear material (SNM) as defined by the Nuclear Regulatory Commission (NRC). SNMs like uranium-235 and plutonium-239 are both fissile isotopes and could be used in nuclear explosives [13].

Different kinds of radiation have different properties. Alpha particles are charged particles which are barely penetrating. While internally hazardous if consumed or breathed in, they are halted by skin or just centimeters of air. Uranium and plutonium are common emitters. Beta particles are electrons from the radioactive nucleus, which are emitted across a wide energy range. Their travel distance through air of a few meters means eyes and skin are at risk, depending on the energy. Carbon-14 and phosphorous-32 are common emitters; plexiglass is an appropriate shield in order to prevent Bremsstrahlung x-ray radiation as the alpha particles pass near the nucleus. Gamma rays are electromagnetic photons from the nucleus of a decaying atom. Traveling tens to hundreds of meters in air, they are highly penetrating and dangerous, requiring lead or dense concrete shielding. Cobalt-60 and cesium-137 are common emitters. X-rays are similar to gamma rays, but are emitted when electrons around the nucleus change valance shells. Neutron emissions are neutrons discharged from the nucleus. Their large mass and size causes them to travel long distances, requiring concrete or water shielding. Plutonium-239 is a common emitter [13]. A simplified diagram of radiation types and shields is given in Figure 1.1 from [1].

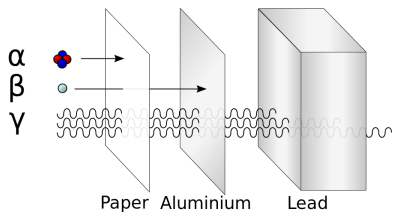


Figure 1.1: Different types of radiation are halted by different shields [1].

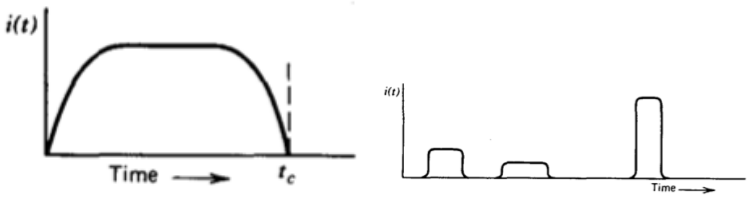
In order to provide safety from or obscure radiation, shielding is used; material properties are carefully selected to produce certain probabilities of interaction. This probability of interaction, the nuclear cross section σ, is a function of the material density η and reactivity ρ of the shield as $\sigma = \eta\rho$. For liquid and solid shields, a higher nuclear cross section is generally desired [14]. Distance can also lower observed radiation; as the inverse square law states, the observed intensity I of radiation emitted by a radioactive point source decreases by a factor of four if the distance d from the source is doubled as $I = \frac{I_0}{d^2}$, where I_0 is the source intensity. Radiation exposure can be drastically modified near a source due to small changes in distance. If an assumption is made that a single material separates the observation point from a source, both the nuclear cross section and distance properties combine. The approximate observed intensity given a source intensity of I_0 is

$$I = \frac{I_0}{d^2} e^{-\sigma d}$$

In order for any detector response to occur, radiation must interact with the detector material. There are multiple mechanisms by which this interaction can occur. Fast electrons and heavy charged particles affect the detector electrons through the coulomb force as they pass, while neutrons, x-rays, and gamma rays catastrophically interact with the detector atoms, producing resultant charged particles. The probability of interaction, called the nuclear cross section, is a function of the density and reactivity of the detector material [14]. These interactions have stopping time timescales in the nanoseconds for liquids and gases, and in the picoseconds for solids, both of which can be thought of as instantaneous. Through the imposition of the electric field as current I in the detector over time t_c, a small amount of electric charge Q is recorded as in Figure 1.2a.

$$\int_0^{t_c} I(t)dt$$

For this discussion a single particle, or quantum, is assumed, but in reality many quanta interact. Over time, more quanta arrive at randomly distributed time intervals according to Poisson statistics, which is shown in Figure 1.2b. Detectors operate in various modes, but pulse counting is common where intensity is more important than the incident energy



(a) A single radiation pulse

(b) Multiple radiation pulses appearing at randomly distributed time intervals

Figure 1.2: Interactions in the detector are recorded as pulses. (Knoll fig 4.1, 4.2)

distribution. If the incidence of events is too high, then current mode, which calculates the time average of current bursts using an ammeter, is used [13].

There are standard types of detectors such as ionization chambers, proportional counters, and Geiger-Müller counters, but this discussion is limited to scintillation detectors coupled with photomultiplier tubes (PMTs), which work on an extensive range of radiation types and have widespread use. These detectors form charge indirectly by recording light intensity over time in the form of discrete pulses.

When scintillation materials are struck by a charged particle from radiation, light is emitted. These materials should have a high efficiency, perform the transfer of kinetic energy to light with linear proportionality, have a short decay time to allow fast pulses, and emit light in visible wavelengths. Finding such a material that prompts fluorescence (visible radiation) but minimizes phosphorescence and delayed fluorescence (lower wavelengths at slower timescales) while preserving these other properties is an exercise in compromise. One example material is the inorganic alkali halide crystal sodium iodide. In semiconductors and insulators, electrons sit in discrete energy bands, staying still in the lower valance band, and moving around in the conduction band. Charged particles energize valance electrons, creating electron holes and elevating the electrons to the conduction band, where they traverse back to an empty hole and release light. However, in a pure crystal this process is not efficient, so impurities are added as activation sites. The electron holes move to the activation sites, which then have a smaller energy gap and release a properly energized photon when an electron drops back from the conduction band to the valance band.

Photomultiplier tubes, seen in Figure 1.3 are required because the scintillation emitted is weak, typically consisting of hundreds of photons, which is not enough to sustain an electrical pulse. Performing this conversion without adding excessive noise is challenging. In the first stage, the photocathode material absorbs photons, adding enough energy to migrate the electron to the surface of the photoemissive material and cause its escape into the photomultiplier internal vacuum. In the second stage, a current is supplied to accelerate the photocathode electrons to the surface of a dynode electrode being supplied with a bias of electrons. With specific materials, the energy of the incident higher energy electron can cause the reemission of two or more electrons from the surface [13].

1.2.2 Machine Learning

This section provides a brief overview to those who are unfamiliar with machine learning, pattern recognition, or artificial intelligence topics, in order to help classify and explain some of the common techniques and algorithms applied in this thesis. First, broad strokes classifications and generalized theory are discussed, and then explanations of specific methods are given.

General Method Types

Humans frequently make immediate estimates about and discriminations between input data. These predictions are often simple to a human, but are difficult to implement on a computer. However, computational power can also allow for a more nuanced interpretation of the data, sometimes at a much higher speed. Machine learning algorithms model data to predict outcomes. The general flow for any machine learning problem is to: obtain data, preprocess it by cleaning or modifying, train a model using a portion of the data, and finally evaluate the model accuracy with the remaining test data [15].

A dataset, as in Figure 1.4, is made up of various components. Each row in the dataset is a sample, with some columns as features, the values used to make the prediction, and one column as the prediction. The samples are divided into two partial sets, one for training, and one for testing. Using the training set, machine learning algorithms attempt to create

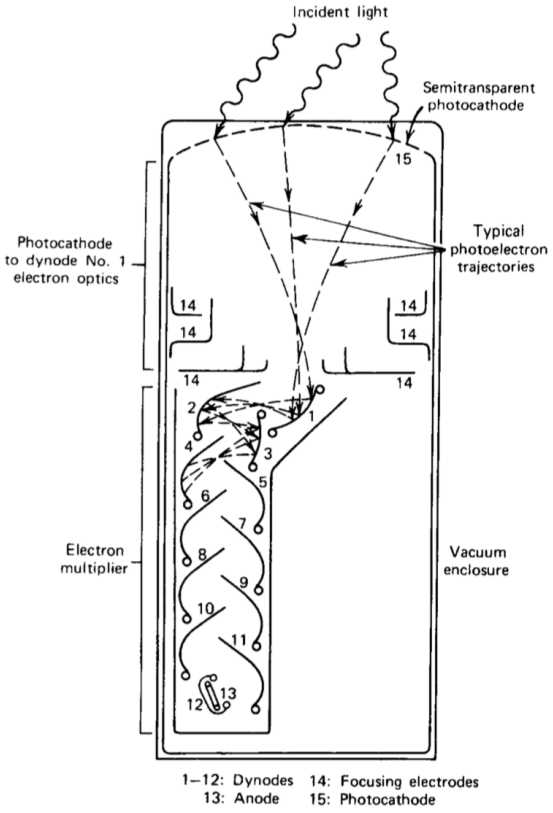


Figure 1.3: A simplified photomultiplier (Knoll fig 9.1)

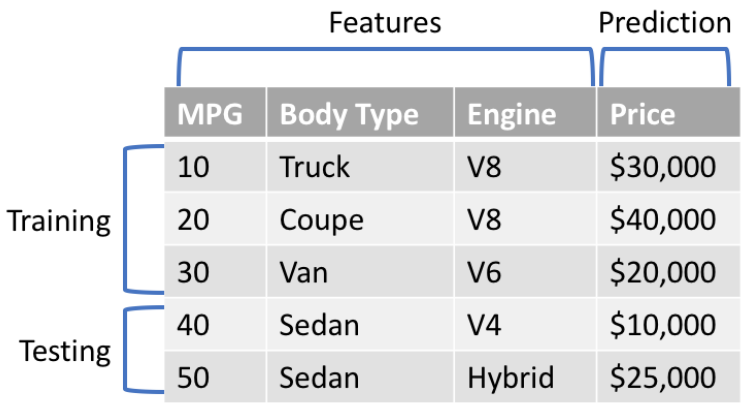


Figure 1.4: An example dataset showing various components.

a statistical model of the data distribution by iteratively and incrementally updating an initial guess. Accuracy is then measured by comparing the true prediction value with the algorithm's guess.

There are a variety of approaches when applying machine learning to a dataset, but generally they can be categorized into a few basic binary groups. In classification problems the goal is to separate data into different categories, while in regression a continuous value prediction is made. In supervised methods, the class of the training data is known, while in unsupervised methods the appropriate classes must be determined. In parametric methods, the probability distribution of the features is assumed, while in nonparametric methods no assumption is made. These categories are discussed further below.

In classification algorithms, a discrete category is assigned to each sample of the data, while in regression algorithms, a continuous prediction is made using the features. For example, given features such as miles per gallon, body style, and engine type, a classification problem might attempt to predict the model of the car, while a regression problem might attempt to predict the price [16].

The method of training using input data can take two forms: either supervised or unsupervised. In supervised training, the classes of training samples are known. This classification of the training data must be performed by a human, a requirement that at the very least takes time, and in the extreme can be challenging. When this human involvement is not tenable, unsupervised classification is appropriate. In unsupervised classification, the

classes of samples are not known, and the number of appropriate classes must be determined. In order to solve these problems, clustering is used based on distance metrics. Between the features of samples is a calculated measure of similarity, such as Euclidean distance. This metric can be used to compare samples, and to find clusters of similar samples. In Figure 1.5 from Analytics Vidhya [2], samples are comprised of income and debt features. If the features are plotted, it is possible to see three separate clusters of individuals. In this case, the discrimination is easy. With more data, more dimensions, and closer clusters, the problem becomes more challenging, and more appropriate for a computer to solve [16].

There are two options for estimating the probability distribution of features: parametric methods and nonparametric methods. In parametric methods, the distribution is assumed to be known; for example a normal (Gaussian) distribution would be appropriate for human heights. In nonparametric methods, there is no assumption of the form of the distribution [16].

Common Algorithms

Maximum A Posteriori (MAP) in a Bayesian setting is a parametric supervised classification method. In Bayesian theory, one calculates the posterior (post-event) probability of an input belonging to a particular class by taking the product of the prior (pre-event) probability and

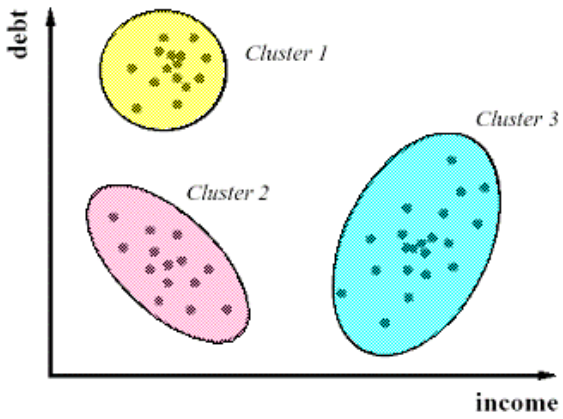


Figure 1.5: A simple unsupervised clustering example from Analytics Vidhya [2].

the likelihood of that event occurring and dividing it by the evidence.

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{p(x)}$$

Methods based on this expression are a popular solution in pattern recognition. Classes are chosen based on the highest a posteriori probability $P(w_j|x)$, calculated using the a priori probability of a given class $P(w_j)$, the evidence $p(x)$, and the probability distribution function for the feature with respect to the given class $p(x|w_j)$. When the prior probabilities for all classes are equal, the method simplifies to Maximum Likelihood Estimation (MLE), another common algorithm [16].

K-Nearest Neighbors (k-NN) is a nonparametric supervised classification method. The motivation for k-NN is to estimate the probability distribution function without the assumption that this probability density has a particular form. Given a point x, we can grow a hypersphere of volume V that encloses k points. Counting the number of samples belonging to class m in that region as k_m, the estimated density is calculated as follows: $p(x|w_m) = \frac{k_m}{n_m V}$ is the probability distribution function, $P(w_m) = \frac{n_m}{n}$ is the prior probability, and $p(x) = \frac{k}{nV}$ is the evidence. The posterior probability is calculated as

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{p(x)}$$

For the sample x, the class with the highest posterior probability is chosen as the decision. To calculate the enclosing volume, multiple metrics can be used for distance. Variations on the Minkowski distance

$$L_k(a,b) = (\Sigma_{i=1}^{d}|a_i - b_i|^k)^{1/k}$$

such as the Euclidean distance at $k = 2$ are most commonly used. As the k value is increased, more and more samples are included in the calculation as the hypersphere volume increases.

K-Means Clustering is a parametric unsupervised classification method. K-Means is a centroid-based clustering technique, where clusters centers are represented by a structure identical to the individual samples. Because k-means is a parametric method, the number of clusters must be given at the start of calculation, a significant drawback. However, the

simplicity of implementation and relatability to the k-nearest-neighbor algorithm makes this a commonly chosen method. The algorithm looks for the optimal cluster placement such that the squared distances to each sample in the cluster is minimized. The algorithm begins by creating arbitrary cluster centroids, and then assigning samples to the nearest centroid. The centroid is then recalculated as the mean of the assigned samples. Samples are reassigned, and if any sample classification has changed, then centroid is recalculated and the process begins again until convergence, when the centroids no longer move [17].

Support Vector Machines (SVMs) are a parametric supervised classification method. For classification using SVMs, the goal is to find the optimal separating hyperplanes between classes by maximizing accuracy and the margin between the discriminating hyperplane and the classes on each side. Consider the two class case, using labels -1 and +1 for the two classes C_t as in Figure 1.6 from [3]. The hyperplane can be defined for class +1 as

$$w^T x^t + b \geq +1$$

and for class -1 as

$$w^T x^t + b \leq -1$$

This +-1 boundary creates the margin that assists with generalization. The discriminant to the margin should be at least some value ρ and is calculated as

$$\frac{C_t(w^T x^t + b)}{||w||} \geq \rho$$

for all classes t. $\rho||w|| = 1$ is fixed to prevent infinite solutions, and to maximize the margin, we minimize $||w||$. For linearly separable problems, this becomes a standard quadratic optimization problem as $\min 1/2||w||^2$ subject to $C_t(w^T x^t + b) \geq +1$ for all classes t. This method's complexity depends on the number of features [17].

Decision Trees are a nonparametric supervised classification method. In decision tree methods, a hierarchical model is used to identify recursive splits in attributes of the data, as in Figure 1.7. Each decision node of a decision tree functions as a split point on a threshold value. A given input is applied to the tree root node, and at each node is tested by a

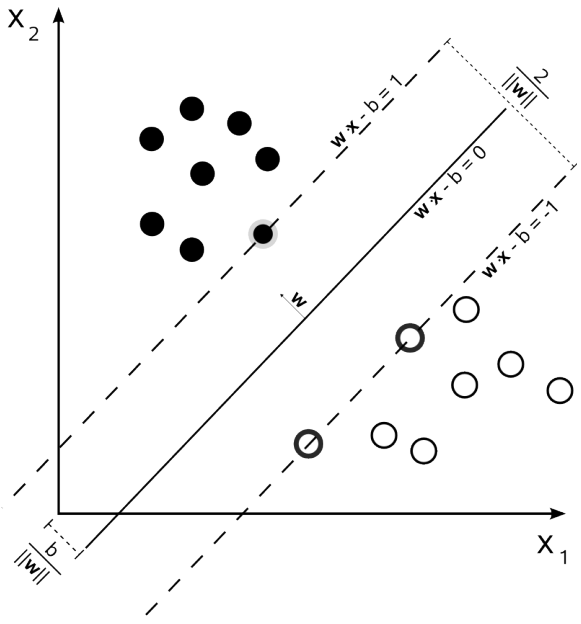


Figure 1.6: A simple 2D separating hyperplane [3].

thresholding function, following a branch based on the result. Leaf nodes indicated regions in space where all instances fall into the same class. Each thresholding function works on a single attribute, sending values less than the threshold to the left node, and sending values greater than or equal to the threshold to the right. For a classification tree, training splits are decided based on an impurity measure. Splits are considered for all attributes and possible thresholds, and impurity is calculated for each, with the lowest impurity being selected. Misclassification error, a common impurity measure, is calculated as $\phi = 1 - \max(p, 1-p)$. If a node is not pure, it should be split to decrease impurity. Splitting stops when a maximum tree depth is reached, a minimum impurity is reached, or when an impurity of zero is reached for a node [17]. In some cases, a single decision tree alone cannot successfully model the data. Ensemble methods boosting and bagging combine multiple trees to produce a more robust model. Bagging trains multiple models on portions of the data, and then averages the result for prediction. Boosting trains multiple models in sequence; the first uses the training data, and subsequent models train on the residuals between the ground truth and the predictions of the previous model.

Multivariate Regression methods are parametric supervised regression methods. In regression, given an input x_i, the output r_i is a numerical value, and a numeric function

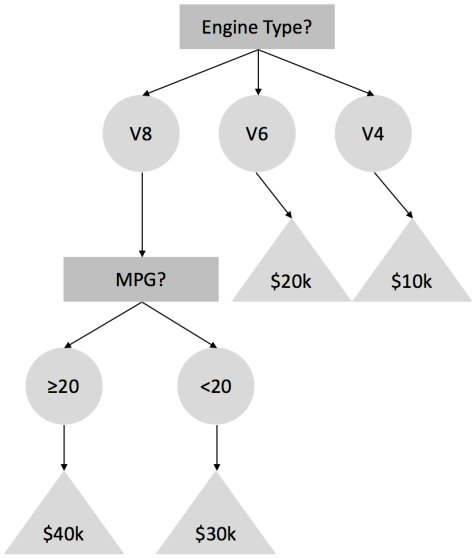


Figure 1.7: A simple decision tree to predict car prices.

$f(x_i)$ should be learned. In machine learning, this function is learned from a set of i training examples with assumed noise ϵ as $r_i = f(x_i) + \epsilon$, and estimated by $g(x_i) = w_0 + w_1 x_i$. The error on the samples is calculated as $E(g(X)) = \frac{1}{N} \sum_{i=1}^{N} [r_i - g(x_i)]^2$ where N is the number of samples. By taking the partial derivatives with respect to the weighting variables, and setting them equal to zero, the minimum can be solved for. In multivariate linear regression, the numeric output $g(\vec{x})$ is produced by a weighted sum of several input variables $x_1, x_2, ..., x_d = \vec{x}$ and noise as

$$g(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_d x_d + \epsilon$$

Now the empirical error on the sample set is

$$E(g(X)) = \frac{1}{N} \sum_{i=1}^{N} [r_i - g(\vec{x}_i)]^2$$

but the same method of taking the partial derivatives with respect to the weights and setting them equal to zero is possible. This results in a set of normal equations which can be written as $X^T X \vec{w} = X^T \vec{r}$, where X is a matrix of all samples $\vec{x}_i$, $\vec{w}$ is a vector of the weights, and $\vec{r}$

is a vector of the real outputs. Solving for the weights becomes as simple as

$$\vec{w} = (X^T X)^{-1} X^T \vec{r}$$

Now the approximation can be calculated by $g(\vec{x}_i) = \vec{x}_i \vec{w}$. This method can also be extended simply for univariate and multivariate polynomial regression. For the univariate case, given an order k for the function, set the variables as $x_1 = x, x_2 = x^2, ..., x_k = x^k$ and solve the same way as before. For the multivariate case, the cross products of variables are also considered as in the function $f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$, causing the transformation of variables as $x_1 = x_1, x_2 = x_2, x_3 = x_1 x_2, x_4 = x_1^2, x_5 = x_2^2$. Again, the final solution for the weights and approximation is the same as in the linear case [17].

Neural Networks and Deep Learning

Artificial Neural Networks (ANNs) are a nonparametric supervised classification method. The motivation for neural networks is to learn higher level functions using combinations of linear perceptrons. Given a set of inputs to a perceptron $(x_1, x_2, ..., x_d)$ along with a bias node of value $b = 1$ and a set of weights $(w_1, w_2, ..., w_d)$ the final value is calculated as

$$(\Sigma_{i=1}^{d} x_i * w_i) - w_b * b$$

where d is the number of features input. Unfortunately, these perceptrons are limited to linear combinations of the inputs, which prevents approximation of functions like XOR, which is not linearly separable. To solve this problem, multiple perceptrons can be connected together, with the output of one perceptron being fed into an activation function before leading to the input of another. These activations functions mimic the behavior of activation in real neurons, and are represented by functions like sigmoid. A perceptron and activation function unit is called a neuron, shown in Figure 1.8. Artificial neural networks vary widely in structure and function.

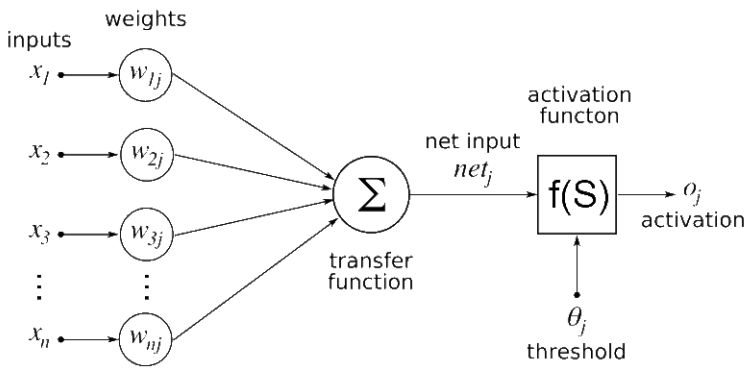


Figure 1.8: An example artificial neuron [4].

With neural networks, shown in Figure 1.9, the challenge is to find weights that minimize the error between the expected value T and true output S in the form

$$E = \frac{1}{2}\Sigma_j(T_j - S(y_j))^2$$

The desired response of every output neuron is known, but there is no way to known the desired responses of the hidden layers. To accomplish this, backpropagation is used, based on gradient descent. The four steps to backpropagation are inputting the patterns, making a forward pass to determine the error, making a backpropagation pass, and updating the weights [16]. In backpropagation, the objective is to calculate the contribution in error change caused by the weights of each layer. Since each weight value in each layer is calculated in the same way, we can consider values as vectors and matrices, and program the same way. The chain rule is used to go back along the chain of equations, which at this point are unknown. First the special case of the output layer is considered.

$$\frac{\delta E}{\delta W[L-1]} = \frac{\delta E}{\delta y}\frac{\delta y}{\delta Z[L-1]}\frac{\delta Z[L-1]}{\delta W[L-1]}$$

The change in error is also considered with respect to the previous layer inputs.

$$\frac{\delta E}{\delta X[L-1]} = \frac{\delta E}{\delta y}\frac{\delta y}{\delta Z[L-1]}\frac{\delta Z[L-1]}{\delta X[L-1]}$$

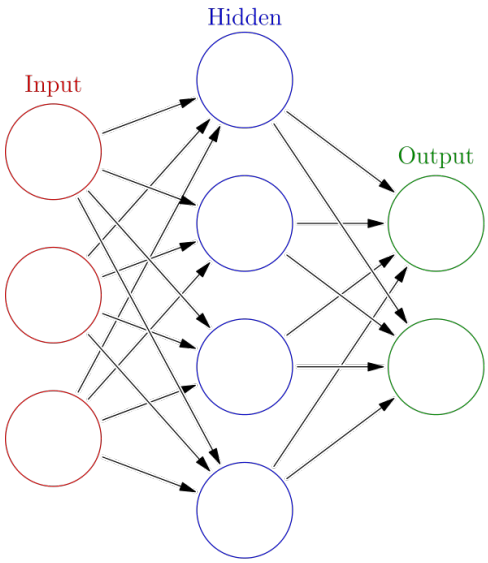


Figure 1.9: An example fully-connected artificial neural network with a single hidden layer [5].

Next, the rest of the layers are considered iteratively from the end of the network to the beginning.

$$\frac{\delta E}{\delta W[l]} = \frac{\delta E}{\delta X[l+1]} \frac{\delta X[l+1]}{\delta Z[l]} \frac{\delta Z[l]}{\delta W[l]}$$

$$\frac{\delta E}{\delta X[l]} = \frac{\delta E}{\delta X[l+1]} \frac{\delta X[l+1]}{\delta Z[l]} \frac{\delta Z[l]}{\delta X[l]}$$

These equations can be applied to any error function, activation function, or output function. Cross entropy error is typically used with various activation functions. Luckily, cross entropy has the nice property of returning the same value for all three output functions in the backpropagation.

$$\frac{\delta E}{\delta Z[L-1]} = y\_ - y$$

Here are the rest of the specific derivative results that can be multiplied as above [17].

$$\frac{\delta X[l+1]}{\delta Z[l]} = logistic\ sigmoid' = f'(Z[l]) = f(Z[l]) * (1.0 - f(Z[l]))$$

$$\frac{\delta Z[l]}{\delta X[l]} = W[l]$$

$$\frac{\delta Z[l]}{\delta W[l]} = X[l]$$

For complex problems, hand designed features by human experts often fall short of capturing the input. In these situations, deep networks can automatically perform feature extraction. Typically artificial neural networks of two or more layers are referred to as deep. As the network iterates from the input to the output, higher level features are extracted by successive layers. Convolutional neural networks (CNNs), a type of neural network optimized for image processing, are discussed in section 1.2.3.

1.2.3 Visual Algorithms

This section provides a brief overview to those who have not learned about digital image processing, object detection, or object tracking. First, image organization and manipulation are discussed, and then common feature extraction methods for detection and tracking are explained.

Images are composed of discretized wave information from the electromagnetic spectrum, ranging from radio waves at a wavelength of 10^3 meters, to gamma waves at a wavelength 10^{-12} meters, with the visible spectrum around 10^{-6} meters. Applications for digital image processing extend across every science and industry, but a few of the most common are gamma ray imaging for nuclear medicine, x-ray imaging for bones, visible light imaging at every scale from satellite photos to microscopic photos, microwave imaging for radar, and radio wave imaging for astronomy [18].

Raw wave information is collected by antennae, but cameras discretize this information into pixels by digitizing the spatial coordinates (sample) and digitizing the intensity amplitude (quantize). This results in a black and white 2D matrix of intensities, with black represented as 0 and white as the highest value. An M x N image with L digital intensities requires b bits as $b = MN \log_2(L)$. Increasing the spatial or intensity resolution requires more storage. In order to work with color images or images with bands like infrared or ultraviolet, additional matrices are added. In this case, a single 3D spatial coordinate is called a voxel. For color images, one popular option is to have each matrix represent the intensity of an additive primary color: red, green, or blue, which allows any color to

be formed by modulating intensities. Surrounding pixels and voxels can be considered by adding or subtracting one from the spatial coordinates, to yield eight or twenty-six neighbors, respectively [18].

One commonly used feature of images is the histogram. Histograms are computed over an image by binning and counting intensity values, which can thought of as a probability distribution over the image. The histogram shape is related to image appearance in a variety of complicated ways, but the feature itself is simple and quick to compute, and is excellent for real time applications. For color images, typically hue saturation intensity (HSI) representation is used for histograms [18].

There are a variety of simple arithmetic, logical, spatial, matrix, and probabilistic operations that can be applied to images, but all are applied using a certain kernel, or small matrix. This kernel can be the entire image, or it could just be the surrounding pixels as discussed above. In order to apply a kernel to an image, the convolution operation is used. Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel, defined as

$$(w \star f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t)$$

where w is the kernel of size M x N such that $a = (M - 1)/2$ and $b = (N - 1)/2$, $f(x, y)$ is the image, and $\star$ is the convolution operator. Using these filters and other methods, it is possible to perform astounding feature detection operations such as point, line, and edge detection and image segmentation [18].

Using the Fourier transform it is possible to convert inputs into the frequency domain. In the case of the continuous 1D Fourier transform

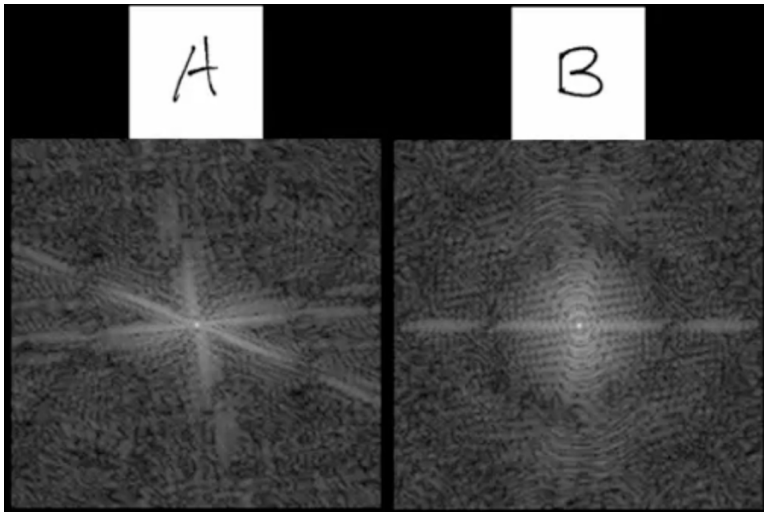
$$F(g(t)) = \int_{-\infty}^{\infty} g(t) e^{-i2\pi f t}$$

where f is frequency continuous variable and t is time, time is converted to the frequency domain. For images, the discrete 2D Fourier transform

$$F(g(x,y)) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x,y) e^{i2\pi(f_x x/M + f_y y/N)}$$

where $g(x,y)$ is an image of size $M * N$ and f_x f_y are the two frequency continuous variables, converts space to the frequency domain. In this domain, new obvious features appear as in Figure 1.10. Additionally, it is possible to perform filtering not on the image itself, but on its Fourier transform. After the filter is applied, the inverse Fourier transform is then used to recreate the original. Noise can be removed by with a low pass filter, and edges can be extracted with a high pass filter. Implementing these equations using brute force methods is of order $\mathcal{O}(MN^2)$, which becomes an issue for even medium sized images. Fortunately, the fast Fourier transform (FFT) reduces this to $\mathcal{O}(MN \log_2(MN))$ [18].

Although automatically extracting features from images is desirable, neural networks face issues with image inputs because pictures are large, and fully-connected layers of neural networks have too many weights. This increases training capacity and requires many training examples. Computationally, the memory and calculation requirement for



(a) A handwritten A **(b)** A handwritten B

Figure 1.10: Two handwritten letters and their Fourier transformations from Michelle Dunn at Swinburne University of Technology [6].

brute force methods is too high. For image tasks, convolutional neural networks (CNNs) are more appropriate. The inherent structure of the convolution operation has more prior knowledge about the domain than standard feedforward neural networks, and the more sparsely connected nature of CNNs means training is less resource intensive. Using CNNs largely reduces the number of weights needing to be trained. CNNs automatically obtain translation-equivariance without replication of weight configurations across the space, and without needing to perfectly center, size normalize, and rotate the data. If the image is shifted, the feature map will be shifted by the same amount, but will otherwise remain constant, providing robustness. By considering the topology of the input, each successive layer can find higher level features, from local to global [19].

A basic CNN from Yann Lecun's 1998 paper is shown in Figure 1.11. In order to execute the convolutional layer, a region of pixels (receptive field) is moved over the input in a certain increment (stride). The convolution is performed for each receptive field and a matching set of weights (kernel) to generate a single value in the next layer (feature map). There is one kernel for each feature map, so that the same feature is detected everywhere in the image. These kernels are the automatically learned feature extractors. Next the feature maps are reduced in spatial resolution by pooling, which increases translational invariance. Pooling is performed by dividing the features maps into small neighborhoods, and replacing every element by a single value such as the average or maximum. Finally, a fully connected layer turns the vectorized final feature maps into standard output predictions [18].

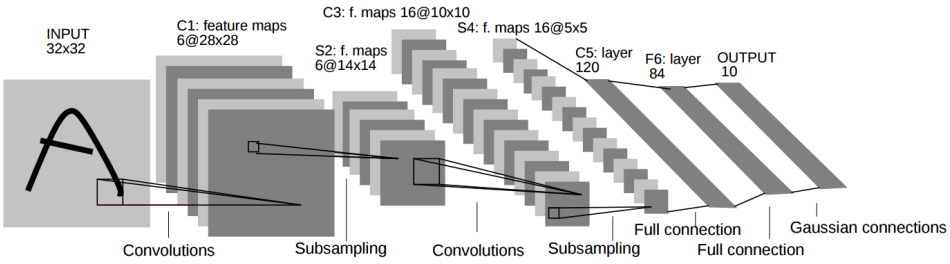


Figure 1.11: The LeNet5 architecture [7].

1.3 Related Works

1.3.1 Nuclear Localization

When assessing an illicit nuclear source, three methods encompass the necessary reconnaissance tasks: detection [20], identification [21], and localization [22] [23] [24]. Detection is sensing the presence of nuclear material, identification is determining the specific radioisotope, and localization is estimating the position. However, a myriad of approaches exist due to the complex interactions of different contexts such as the environment, mounting platform, sources, and detector. In this thesis, a step is taken towards localizing a moving source by estimating the relative source direction in real time. First, these variables are discussed, and then various radioisotope localization and direction estimation method trends are explained.

Although nuclear detection tasks must be carried out in a variety of diverse environments, one system does not need to combat all threats. Assumptions are made for each detector to target specific contexts. The main discriminator between settings is urban [24] [25] versus rural or undeveloped [20]. In urban environments, topography is dense and complex, resulting in diverse absorption and reflectance properties. Crowds and traffic change constantly. NORM sources such as potassium-40 in concrete are more common. In this context, localization perspectives must vary between two-dimensional and three-dimensional localization depending on the the platform. In rural environments, topographies are more open, with shielding provided by plant life. Less NORM sources are present, but the lack of shielding resulting from the open setting allows radiation to travel farther. Here, the topography is more stable, and two-dimensional localization is most appropriate. In this thesis, the system is built to operate in various environments, but can handle the stresses of crowded urban contexts.

As the size, weight, and power requirements for detectors have reduced, the range of mounting platforms has expanded. Stationary [20] detectors such as portal detectors used at shipping yards and border crossing are excellent as monitors. By assuming location in containerized arrivals and departures, more accuracy can be achieved in detection and identification. Aerial [25] [26] detectors such as helicopter mounted systems are optimal for

detection over a large area, but provide unique high speed challenges. Additionally, large areas introduce more confounding sources, and often require exhaustive search or a ground based follow up to achieve acceptable accuracy. Automotive detectors are driven in an area depending upon the capabilities of the vehicle used, focusing on a small area to minimize confounding sources. Unmanned Aerial Vehicle (UAV) [23] [27] and pedestrian detectors have the strictest limits on size and weight, but are optimal for precise follow up in a small area with few confounding sources. This thesis focuses on optimizing for a pedestrian platform.

Various tracking scenarios exist for source-detector pairings. Input is received from a single detector, a number of detectors in a single unit [25], or a network of multiple detectors [24]. These systems can track one or more radioisotopes [21] at the same time, in addition to ignoring background radiation sources. Across different environments the source or detector can be stationary or traveling erratically. For this thesis, the relevant detector components are sodium iodide (NaI) scintillators [20], Cs2LiYC16:Ce (CLYC) scintillators [28], photomultiplier tubes (PMTs), silicon photomultipliers (SiPMs) [29], and digitizers.

Input reconstruction is inherently difficult for nuclear detection tasks. Source emissions are dependent upon random statistics rather than set frequencies, and small variations in environment result in large variations in the myriad occlusion and scattering events possible. When source models are posed with more properties than can be uniquely determined by the observations, sources are rendered unidentifiable, or may be misidentified. In the case of stationary detectors, where the environment is assumed to be known, modeling strategies can be used to recover source properties. With Monte Carlo methods [30], individual particle interactions can be simulated for a set area, and with the Boltzmann transport equation [22], statistical behavior is used to approximate the radiation transport model. Simplifications like assuming only uncollided particles reach the detector [24] are used to simplify the calculation, but many computer resources are required in order to evaluate the various combinations of models. For moving detectors, the challenge increases, as a known environment can no longer be assumed. Instead, field mapping is used to produce a map of the radiation intensity [23] or the likelihood for a given position [22]. In either case, common methods to perform parameter estimation include direct Bayesian applications [25] and maximum likelihood estimators [31]. In this thesis, where both the detector and source can move,

there are many source possibilities given a set of observations. Rather than posing the localization as estimating absolute position and intensity, a simplified model is used with a single input parameter, the angle of the source relative to the detector. Regression techniques are applied towards discovering the most probable angle, and Bayesian updating is performed using the final angle and visual position data to track sources in real time. This directionality representation is well posed and applicable to the self shielding, directionally aware detector setup used.

By arranging detectors to produce self-occlusions, the angular resolution of a detector array is improved. This requires deliberate orientation and positioning of each detector. This allows a variety of approaches which rely on the relative difference in observed counts between detectors. In [32], a fuzzy logic algorithm based on gross count response converts low, medium, and high intensities into a 360 degree directional bearing. This segments the bearing into 16, 22.5 degree portions for estimation, which allows predictions to be made on a moving source over 0.5 seconds at a certain, unspecified distance. In [33], angular resolutions of one degree are possible using a 100 microcurie source within three meters over a certain, unspecified period of time. This method finds a linear relationship between detector asymmetry and source angle.

For this thesis, radioisotope direction estimation is performed for a single moving radioactive source. This estimator predicts a single direction after each real time interval; during each interval the motion of the source and detector is assumed to be minimal. The directionally aware detector consists of two CLYC detectors with photomultiplier tubes and two sodium iodide detectors with silicon photomultipliers, where the exact physical detector setup is known, including the pedestrian body. This design [34] comes from work by our research group.

1.3.2 Object Detection

Visual detection is the problem of determining whether an object belonging to a specified class is present, and localizing it in the image. Typically this location is represented by a bounding box around the object. Historically, detectors were trained for face detection

[35]. Currently, because applications include robotic interaction, autonomous driving, and assistive devices, algorithms attempt low latency predictions for real time use.

Detection is a field with a high amount of research depth, and a long history. This is a result of the significant challenges that detection presents. Individual objects in images are subject to occlusions, illuminations, pose variations, variable resolutions, and object similarity. Since detection tasks are highly dependent on contextual information, it is important that detection datasets properly represent objects in their natural environments. This becomes a difficulty for detection as labeling localization information is more challenging than a simple class definition, as in the case of classification datasets. Finally, different tasks require a variety of objects to be learned by a single model; whether these objects are highly similar or widely variable is unknown, and challenges the model capacity. Even intraclass variability can be high due to object rotations and deformations.

In the early days of object detection, complicated pipelines using handcrafted features were state-of-the-art. Typically methods relied on a dense sliding window [36] or sparse region proposal method like selective search [37] to produce candidate regions, with an image pyramid to extract features and make predictions; then post processing was required to eliminate duplicates and clean bounding box borders. This led to a stagnation in accuracy improvement.

Newer methods like R-CNN [38] continue to use region proposal methods such as selective search, but remove the complicated pipeline in favor of a CNN for feature extraction. However, one CNN pass is required for each object proposal, without shared computation. R-CNN additionally solves the scarce training data issue by pretraining the CNN weights using an image classification dataset. Instead of classification, OverFeat [39] instead performs detection using localization by a fully-connected layer on candidate regions which assume a single object. DeepMultiBox [40] improves candidate recommendation using a deep neural network to predict multiple class-agnostic boxes.

Rather than adopting these two stage methods for region proposal followed by detection, SSD [41] dispenses with region proposal, instead assuming default boxes of set size spaced around CNN features. YOLO [42] similarly achieves impressive real time performance by running one CNN on the full image context in one pass. This approach also provides

greater generalization than methods which only observe local information. DSSD [43] adds deconvolution operations to solve this problem with context, while ION [44] propagates contextual information and spatial information from outside the region of interest using recurrent neural networks and skip connections, respectively.

In the most modern methods there is conflict between single stage [45] and two stage [46] [47] detectors, with both increasing efficiency by sharing computation between operations. Fast R-CNN [48] improves the training stage to jointly learn to classify object proposals and refine their spatial locations, but still applies a costly subnetwork on every region. Faster R-CNN [49] integrates a region proposal network into the detection CNN to share computation. YOLOv2 adopts another version of the default boxes called anchor boxes; specifically sized boxes are assumed and then adjusted by the network, rather than predicting all box coordinates directly. Additionally, in order to trade off speed and accuracy without retraining, YOLOv2 trains on images scaled to varying resolutions. YOLOv3 [9] also adopts the new trend of predicting at various scales through the use of a pyramid network [47].

Recently, concerns over the realism of bounding box IoU accuracy measures [9] have led some researchers to attempt image segmentation tasks. Mask R-CNN [50] follows this trend, adding a third branch to Faster R-CNN for predicting an image mask. Perhaps in the future the bounding box will be abandoned in favor of more realistic metrics.

1.3.3 Object Tracking

Visual tracking is a decades old, fundamental computer vision problem, with many applications such as robotics, augmented reality, and surveillance. In the first frame of a sequence video, a previously unknown target is specified by a bounding box. The tracking algorithm must then identify the target in subsequent frames by specifying new bounding boxes 1.12. In order to solve this problem, trackers must model the object in the first video frame, and then continue to recognize the object or differentiate it from the background in order to localize it. Ideally, this process would occur at or above real time while remaining robust to changes in the target's appearance.

Over the past decades, this problem has been investigated from a variety of research angles, has multiple benchmark datasets, and makes up a significant portion of submissions



Figure 1.12: Trackers attempt to place an estimated bounding box A_T with as large an overlap (yellow) as possible with the ground truth A_G. (Background image from [8])

to top-tier computer vision and machine learning conferences. This research depth is possible because challenges arise due to object deformation, variations in lighting, and occlusion. Learning an object representation for the tracking challenge is difficult. Individual video sequences have arbitrary characteristics and contain a variety of objects, each with a different shape, appearance, and movement style. Objects that appear in the background of one sequence could be the target in the next. Additionally, confusing changes such as deformation, motion blur, camera movement, and illuminations can occur across frames in a single sequence. Objects can be transparent, look similar to clutter in the background, and move erratically. This variability ensures that simply learning a model from the initial frame will not work.

Visual tracking has many approaches which are too broad to cover here, but can generally be divided into a few categories. Although these boundaries can be crossed, methods include generative, discriminative, part-based, combination, and deep convolutional neural network based methods. Across all of these methods an implicit motion model is assumed as trackers search a candidate space surrounding the previous frame, but state-of-the-art results can be obtained by searching the entire frame [51]. Over the past few years, various methods have diminished or increased in popularity; this is observable in the submitted trackers to the Visual Object Tracking (VOT) Challenge in Figure 1.13.

Generative and discriminative models are the traditional ways of separating method types. Generative methods [52] [53] [54] [55] [56] model the target appearance and compare

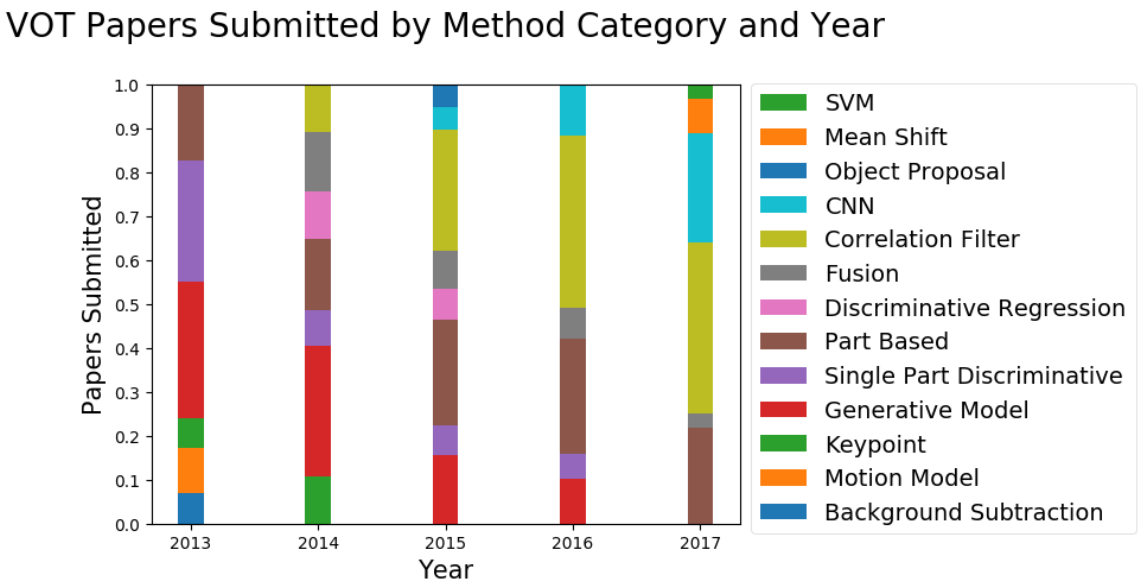


Figure 1.13: Papers submitted indicates fraction of total papers. Over the past five years, CNN and correlation filter methods have become dominant [8].

this representation to search regions to find the best match, while discriminative methods [57] [58] [51] [59] seek to differentiate the target from the background in a binary classification.

Each algorithm has intrinsic focuses and weaknesses, leading some trackers to fuse approaches into a unified system. This fusion can occur at the feature level [60] [61], or the results of multiple trackers can be fused [62].

Part based trackers that infer the shape and position of individual target parts assist in solving issues with shape deformation and partial occlusions occur when modeling the entire target. As in whole target modeling, the maximum likelihood among search locations represents the target, but here a larger number of parameters must be estimated. Various methods exist in this framework, including keypoint based [63] [64], flock of trackers [65] [66] [67], and mean shift [68].

In recent years, correlation filters have become increasingly popular in tracker models due to their speed, allowing for object tracking in real time. The speed of correlation filters comes from its use of Fourier domain formulation. The Fourier domain formulation uses the Fast Fourier Transform (FFT) and quick matrix operations to solve the large ridge regression problem quickly [69]. For example, in VOT2016, twenty-six trackers used correlation filters, correlation filters with color output, or combinations of a correlation filter and a CNN [70].

Trackers that are simply variations of a correlation filter include SRDCF [71], SWCF [72], DSST [73], STC [74], DFST [75], and sKCF [76]. Trackers that use correlation filters with color output include Staple [77] and ACT [78]. Trackers that apply CNN feature detection to correlation filters include C-COT [79], MKCF [80], and HCF [81]. VOT 2017 introduced a real time competition, which CSR-DCF [82] won.

Over the past few years, convolutional neural networks have produced tremendous results in computer vision tasks through advancements in deep architectures [19] [83] [84] [85] [86]. Rather than hand-crafting features, CNNs find them automatically. Features from lower layers contain more spatial information, while higher layer features extract more contextual information. Despite success across research areas like object classification [83] [85] [86], object detection [42] [38], and image restoration [87], where CNNs strongly outperform traditional methods, state-of-the-art methods in visual tracking have not been dominated by the same trend. These algorithms train on enormous datasets like ImageNet [88] to produce models accurate enough to outperform humans, but in the object tracking domain data scarcity and variability pose a large challenge. R-CNN [38] proposes a solution to this for object recognition by pretraining on a related dataset and then fine-tuning on the real dataset; this method has been replicated for visual tracking [89] [90], with large improvements gained from pretraining on ImageNet. State-of-the-art CNN based methods increase in number every year, either as core modeling systems or as feature representation learning algorithms.

The most dominant trend for CNN based methods is pretraining a backbone network offline using a detection dataset. MDNET [91] and SO-DLT [89] use ImageNet, and all but one of the top 10 methods from VOT 2017 applies CNN features pretrained on a detection dataset. SIAMFC [10] performs more advanced offline training, focusing on recognizing objects given in one image in a second, larger image. Often methods will use more than one CNN. During online training, TCNN [92] keeps multiple copies of an incrementally trained network in a tree form, while MLDF (based on [93]) and DNT [94] use portions of the same network to capture low level spatial features and high level contextual features. LSART [95], the VOT 2017 winner is a part based approach to CNN, forcing spatial regulation in order to learn separate filters for the decomposed target.

As the current trends continue, methods using hand-crafted features represent less of a majority portion of the state-of-the-art. Each target model must be generated online using a variety of features. Hand-crafted features like HSV color histogram, LBP texture features, HOG edge features, and color name are used in traditional description methods, each with its own shortcomings due to sequence variation. One recent method [96] explicitly switches from edge to color features to minimize these shortcomings on a sequence to sequence basis. In our own research group, a multimodal fusion method by Andrew August and Zhifei Zhang formulates the relation between modalities in physical space to the embedded space of sparse codes as a tree-based hierarchy. This leads to a hierarchical appearance modeling that is able to capture multiple levels of cross-modality correlations while prohibiting misleading co-adaptations between data representations.

1.4 Contributions

The contribution of this thesis is threefold. First, a generalized directional estimator is proposed. Second, two state-of-the-art visual object detection and visual object tracking methods are combined into a single tracking algorithm. Third, those outputs are fused to produce a real time radioisotope tracking algorithm.

Typically localization methods assume a stationary or moving detector setup with the goal of accurately identifying the coordinates of a stationary source. This thesis takes a large intermediary step towards localizing a moving source, assisted by advanced object tracking techniques. The proposed algorithm can estimate carrier likelihood for objects in its field of view, and is designed to assist a pedestrian agent wearing a backpack detector.

Some smaller contributions of this thesis also benefit the IDEAS for WIND project. Lab data using the detector setup [34] designed by the group was recorded, with help from Sean Alcorn and Callie Goetz. As a result of the visual tracking investigation process, resources for finding and evaluating future state-of-the-art methods were recorded. A basic library was created as a basepoint for further iteration on this prototype.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 describes an algorithm for estimating source directionality and provides accuracy and other results across simulated and lab recorded data. Chapter 3 presents a generalized multi-target tracking algorithm using state-of-the-art methods in object detection and tracking. Results of the individual and combined methods are shown, along with details of the datasets and metrics used to evaluate the chosen algorithms. Chapter 4 specifies the details of a prototype algorithm for real time radioisotope tracking. Chapter 5 summarizes the achievements of the work, and reviews the next steps in terms of future planned and suggested tasks.

Chapter 2

Nuclear Directionality

In this chapter, information and results related to nuclear directionality estimation are presented. A flow path from detector output to angular prediction is proposed, and then specifics are detailed along with evaluation results. This starting algorithm is evaluated on a new dataset recorded under lab settings, and experiments to capture basic dataset features are recorded. This flow path is used in the final system in order to track the source angle relative to the detector.

2.1 Algorithm Design

Standard tasks using a nuclear detector include detection, identification, and localization, in order to determine if a source exists, identify the isotope, and estimate a location. Across all these methods confusing naturally occurring radioactive materials (NORM), medical sources, and other nuisance sources obscure the radiological isotopes of interest. In the case of this thesis, design focuses on a single, possibly moving source and a single, moving detector on a pedestrian platform, both in environments with complex local topography. Rather than trying to accurately localize a nonmoving source, this thesis is focused on assisting an agent by estimating direction in real time.

In order to extract the angular position of a single moving source, several operations are required. First, a detector setup records every gamma interaction as an energy pulse. Next, these pulses are converted to ratios of count rates for each separate detector. Then, regression

methods estimate the original angle to produce a likelihood distribution; this distribution is fused with visual object tracking results using Bayesian inference.

Before obtaining a standard dataset on which to perform simple regression techniques, many recording and preprocessing procedures are necessary. First, a source must emit multiple gamma ray pulses. Some of these interact with the scintillation crystals present in the self shielding, directionally aware detector setup. In the case of this detector setup, interactions occur in two CLYC detectors and two sodium iodide detectors. Photomultipliers then pick up the photons emitted by the scintillators and convert the pulses into electricity. For this thesis, photomultiplier tubes and silicon photomultipliers are connected to the CLYC and sodium iodide detectors, respectively. These electrical pulses are cleaned and the area under the pulse is integrated automatically by a digitizer to acquire the total energy of an event. This event is cutoff at some maximum value. Over time, these readings provide a list of matching timestamps and energy readings for each individual detector. Counting the number of pulses over four seconds, experimentally determined as in section 2.2.4 to be the optimal period of time, produces the count rate. In order to standardize the data for various relative source strengths due to distance and absolute source strength, the count rates are normalized across the detectors using individual ratios, determined to be the best method in section 2.2.4. This results in six floating point features corresponding to a single angular input. The number of features is modified based on the detector setup, but otherwise these preprocessing steps produce detector-ambiguous data for regression. The detector design [34] comes from work by our research group. These preprocessing steps are based on work by Carl Britt.

In order to optimize for the minimal energy requirements and varying environment of the pedestrian setup, the simplest possible model with the fewest parameters is assumed in the form of a well posed directionality estimate. The source is represented as a single parameter, the angle, which is interpreted as a point source along a line due to the operative scales in the tens of meters. This model assumes most detections are uncollided gammas emitted by the source. However, as the environment changes, unpredictable error is sure to occur due to scattering and occlusion of gamma rays before they reach the detector. Rather than struggling to recognize these aberrations, a single model of the error is determined as

a Gaussian angle likelihood distribution around the regression-estimated angle. This error is then minimized by coupling relative agent and tracked object movement over time as the measurements progress. Instead of being constrained by a single environment, this model allows the focus to shift to hunting of specific detected and identified isotopes isolated from background radiation. The approximate likelihood distribution is determined in section 2.2.4. Gradient boosting regression is chosen to minimize the size of this distribution as described in 2.2.4.

2.2 Results

2.2.1 Simulated and Real Detector Setups

In order to validate the directionality approach, a dataset of simulated and lab data with which to evaluate prospective methods was collected using the settings given in Table 2.1.

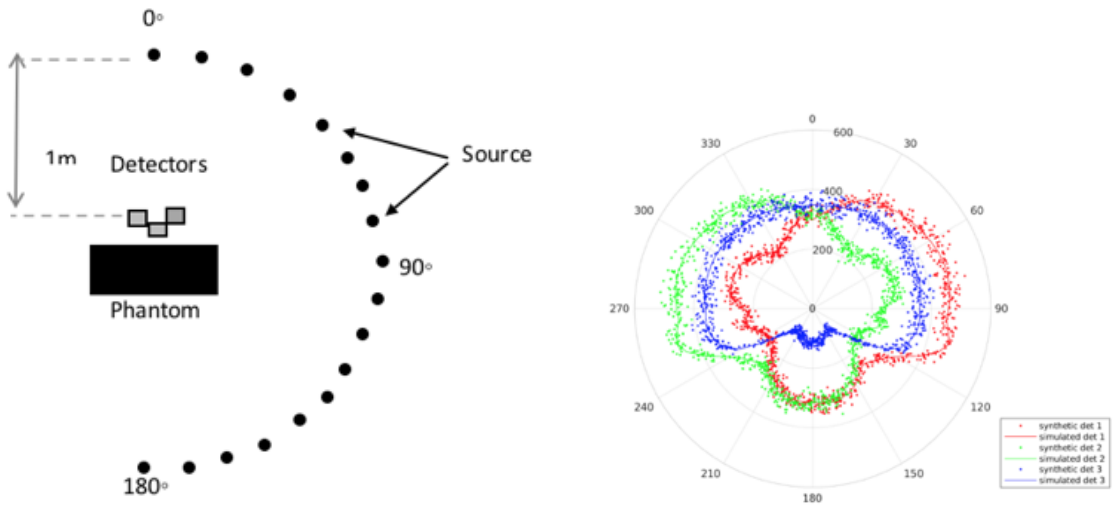
The simulated data is from Carl Britt, produced in MCNP6, employing the specifications given in ANSI 42.53-2013 for a backpack-based radiation-detection system. The setup consists of three right perpendicular parallelepiped (RPP) CsI(Na) detectors (CsI =4.5 g/cm^3) arranged behind a humanoid phantom as in Figure 2.1a. Each detector has crystal mass of 1 kg, a threshold energy of 60 keV, and dimensions of 5.08x5.08x8.59 cm, where the final dimension is the height. A single 200 μCi Cs-137 source which emits gamma photons at 662 keV was used. Thirty-seven simulations were performed - one for each source-angle from 0 to 180 in increments of 5, all at a distance of one meter. The ratios of total counts

Table 2.1: The synthetic and laboratory settings explore different experimental variables.

	Synthetic Data	Lab Data
Source Strength	200 μCi	0.3, 8.15, 9.91, 18.06 μCi
Count Time	1 second	0.25, 0.5, ..., 4, 8 seconds
Distance	1 meter	0.53, 0.91 meters
Isotope	Cesium-137	Cesium-137/Cobalt-60
Background Radiation	No	Yes

for these simulations were calculated. Counts at internal angles were simulated using linear interpolation and Poisson noise, shown in Figure 2.1b.

Lab data was collected using a detector setup [34] consisting of two 32 keV sodium iodides (NaI(Tl)) with silicon photomultipliers (SiPMs) and two 100 keV CLYCs with photomultiplier tubes (PMTs) connected to a CAEN DT5720 digitizer. The combined CLYCs and PMTs measure 54x54x152 mm and were connected end to end, separated by an aluminum plate. Each combined sodium iodide and SiPM measured 54x54x152 mm and were individually placed in the detector setup on each side of the CLYC detectors. The detectors were powered by a lithium ion battery and are connected to a Dell Workstation 5520 laptop. All pieces were held together by a plastic polymer frame, as in Figure 2.2a, inside an internal frame backpack. This backpack was carried by a humanoid phantom that approximates the human body, comprised of varying densities of plastic that simulate bones, organs, and tissues as in Figure 2.2b. For this dataset, the suite of variables investigated include the isotopes, distances, and strengths in an attempt to measure expected performance variation across different algorithms. For each combination of variables, data was recorded for five minutes. Figure 2.3 shows an example of the counts resulting from a single experimental setup.



(a) Detector arrangement **(b)** Example data

Figure 2.1: This is an example of simulated data when recorded in the three detector arrangement. This image was produced by Carl Britt.



(a) The detector setup **(b)** The experimental apparatus

Figure 2.2: The detector setup and experimental apparatus used to record the training and evaluation dataset.

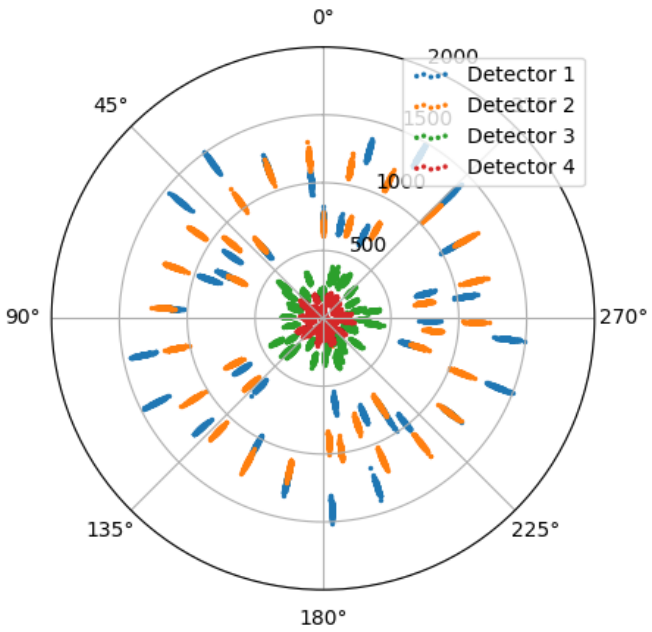


Figure 2.3: Counts over one second intervals for the 8.15 μCi cesium-137 source at 0.53 meters.

2.2.2 Simulation Data

Using the simulated data, various standard scikit-learn regression techniques are evaluated. Each algorithm is run using the default settings with ten-fold cross validation using 100000 data points. In order to measure success, root mean square error is calculated and presented in Table 2.2. Random Forest regression achieved the lowest average error at seven degrees, but this average is not expected on real data, as the noise will increase with the addition of background radiation.

2.2.3 Lab Data

2.2.4 Algorithm Design

Although absolute pulse counts for each detector vary as a function of source strength and distance, the ratios of pulse counts between detectors remain consistent. This can be observed in Table 2.3, which shows the results of three preprocessing methods: total counts (d_1, d_2, etc.), individual detector ratio combinations (d_1/d_2, d_2/d_4, etc.), and ratio to the sum of counts of the other detectors ($d_1/(d_2+d_3+d_4)$). Random forest regression is used to generate

the predictions, with a four second count recording time used. The table is representative of the trends observed in the rest of the data: the lowest errors are produced by the individual detector ratios.

Due to the the probabilistic nature of radiation events, over a longer period of time the detector will observe count statistics more representative of the given source. This leads to greater directional accuracy as observed in Table 2.4. However, since the desired end result is fusion in real time, the recording time is limited.

The rmse errors shown in Table 2.5 come from multiple experimental setups. Across all of these setups, gradient boosting regression consisently appears in the lowest three error values. As a result of this, gradient boosting is used for variable investigation and in the final prototype. These accuracies are lower than the synthetic data, both because of the addition of background radiation, and because of the lower relative source strengths. In the lab setting, this background radiation can easily be removed, but in a real world application it is more challenging.

The distribution of predictions using gradient boosting across all four second counts is shown in Figure 2.4. A Gaussian distribution is fit to this data with a mean of 0.2 and a variance of 51.3. This matches the overall root mean square error calculated of 51.3. However, with greater relative source strengths this error decreases, as seen in Figure 2.5.

Experimental Variables

Regression using ratios can account for the possible experimental variables of source strength and distance, but not source material. This means that one and only one model needs to be trained per source material. This can be observed in Tables 2.6, 2.7, and 2.8, which show the results of training on one experimental setup, and testing on another. For each table, 60 percent training and 40 percent testing data is used for the single experimental setups, while 100 percent training and 100 percent testing data is used when changing experimental setups. Gradient boosting with four second count times is used to generate the predictions for all tables.

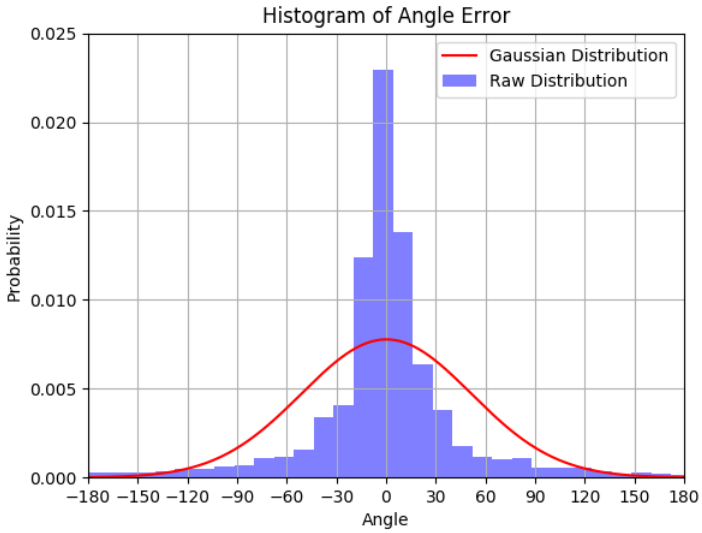


Figure 2.4: The red line is the Gaussian calculated using the mean and variance of all prediction data from gradient boosting.

Table 2.2: This table shows the resulting error for each standard regression algorithm using the simulated data.

Regression Method	RMSE
Random Forest Regression	7°
K Nearest Neighbors Regression	8°
Gradient Boosting Regression	10°
Decision Tree Regression	11°
Logistic Regression	19°
Support Vector Regression	20°
AdaBoost Regression	22°
Multi-Layer Perceptron Regression	38°
Stochastic Gradient Descent Regression	42°
Linear Regression	43°
Bayesian Ridge Regression	45°

Table 2.3: Training on an 8.15 μCi Cs-137 source and testing on an 18.06 μCi Cs-137 source at 0.91 meters using three different ratio preprocessing techniques shows that ratios are necessary for the algorithm to generalize to different relative source strengths. Counts are recorded for four seconds, and random forest regression is used for the predictions.

Ratio Method	RMSE
Total Counts	92.5°
Individual Ratios	48.1°
Sum Ratios	48.8°

Table 2.4: As the counting time increases, the error decreases. These are the results at 0.91 meters using an 18.06 μCi Cs-137 source. Random forest regression is used for prediction given individual ratio features.

Counting Period (seconds)	RMSE
0.25	61.8°
0.5	56.1°
1.0	49.1°
2.0	38.8°
4.0	27.7°
8.0	26.6°

Table 2.5: Five experimental data setups are used to evaluate various regression methods (A: Cs-137/8.15 μCi/0.53m, B: Cs-137/8.15 μCi/0.91m, C: Cs-137/9.91 μCi/0.91m, D: Cs-137/18.06 μCi/0.91m, and E: Co-60/0.31 μCi/0.91m). The lowest three rmse values for each setup are highlighted in red.

Regression Method	A	B	C	D	E
Linear	60.7°	58.5°	60.0°	51.4°	90.2°
Support Vector	54.6°	59.2°	58.4°	55.2°	95.7°
Decision Tree	17.8°	59.4°	57.4°	34.5°	128.3°
Gradient Boosting	16.2°	46.9°	40.7°	29.7°	90.8°
K Nearest Neighbors	11.2°	51.8°	41.9°	29.7°	97.6°
Logistic	34.8°	66.2°	55.5°	48.8°	107.5°
Gradient Descent	74.7°	180.0°	76.6°	53.7°	180.0°
Bayesian Ridge	60.7°	58.7°	59.4°	51.4°	90.1°
Random Forest	12.5°	49.5°	40.9°	28.1°	97.8°
Multi-Layer Perceptron	66.5°	84.6°	78.3°	59.5°	101.8°
AdaBoost	17.8°	48.9°	45.4°	42.7°	91.5°

Table 2.6: The error increases when changing the source material between training and testing. Each setup is recorded at 0.91 meters. The error for Cobalt 60 is much higher because the source strength of the Cobalt source is 0.31 μCi, while for Cesium it is 8.15 μCi. It appears that using ratios also reduces the error when switching source materials, but this is likely due to compensating for the large difference in source strengths of these two experimental setups.

Experimental Setup	Individual Ratio Testing Error	Total Count Testing Error
Cesium 137	46.9°	42.8°
Cobalt 60	90.7°	91.1°
Cesium 137 to Cobalt 60	93.77°	128.9°
Cobalt 60 to Cesium 137	77.6°	181.0°

Table 2.7: Although the error increases when changing the distance between training and testing, using ratios reduces the difference in error. Each setup is recorded at 0.815 μCi.

Experimental Setup	Individual Ratio Testing Error	Total Count Testing Error
0.53 m	16.2°	10.5°
0.91 m	46.9°	42.8°
0.53 m to 0.91 m	64.9°	129.5°
0.91 m to 0.53 m	39.2°	89.1°

Table 2.8: Although the error increases when changing the source strength between training and testing, using ratios reduces the difference in error. Each setup is recorded at 0.91 meters.

Experimental Setup	Individual Ratio Testing Error	Total Count Testing Error
8.15 μCi	46.9°	42.9°
18.06 μCi	29.7°	28.1°
8.15 μCi to 18.06 μCi	48.1°	91.1°
18.06 μCi to 8.15 μCi	77.9°	93.0°

Chapter 3

Visual Object Tracking

In this chapter, information and results related to visual object detection and tracking algorithms are presented. Relevant datasets and metrics are explained before reasoning is given for the choice of algorithm. Additionally, performances on separate datasets relevant to the system use case are demonstrated, including combined performance. These algorithms are used in the final system in order to visually track the position of objects that could be carrying a nuclear source.

3.1 Object Detection

This section provides both a general guide to selecting an appropriate object detection algorithm for this thesis, and also explains the current choice, YOLOv3 [9]. There have been and will continue to be rapid advances in detection efforts over the course of the WIND project; additionally, while the current goals and challenges are known, future design decisions may alter the best choice of algorithm. First, relevant object detection datasets are described, and metrics for comparison are discussed. Next, the case for YOLOv3 is detailed, and an explanation of the algorithm is provided. The results of running this method on a separate challenge dataset are displayed in section 3.3.2

3.1.1 Datasets

The Common Objects in COntext (COCO) [97]; Pattern Analysis, Statistical Modeling, and Computational Learning (PASCAL) Visual Object Classes (VOC) Challenge [98]; and ImageNet [88] datasets are well-cited databases for object detection. COCO consists of 330,000 images labeled for captioning and image segmentation. There are 80 'object categories' with clear boundaries, 91 'stuff' categories with no clear boundaries, and 1.5 million object instances. These images were selected for their natural context, to facilitate detection and segmentation. PASCAL VOC consists of 11,000 images obtained from the Flickr website. These were labeled for classification and image segmentation, with 27,000 object instances. The challenge operated from 2005 to 2012, focusing on object detection in natural images. ImageNet consists of 1.5 million images annotated via a large-scale crowdsourced system. There are 1000 object categories, including many fine-grained differences. Across the spectrum of images are various scales, clutters, and textures, all of which add difficulty to the corresponding challenge run since 2010. These datasets are an appropriate starting place for future algorithm search.

3.1.2 Metrics

Detection algorithms are evaluated by multiple metrics representing accuracy. For a given image, the area of overlap between the ground truth bounding box A_G and a detector's estimated bounding box A_D measures detection accuracy, called intersection over union (IoU). In each image, the accuracy ϕ is calculated as

$$\phi = \frac{A_G \cap A_D}{A_G \cup A_D}$$

Average precision (AP) is significantly more involved. To start, precision is calculated as $TP/(TP + FP)$ and recall is calculated as $TP/(TP + FN)$ where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. Consider an image with six objects. After sorting the top five detection results by confidence, predictions two and five are missed such that the IoU is less than 0.5. This means that the top five predictions have a precision of 3/5 and a recall of 3/6. For the other top four, top

six, etc predictions, the recall value will vary from 0 to 1.0. In order to calculate AP, the average value of all of the maximum precisions stepping by 0.1 from 0 to 1.0 recall is taken. The maximum precision for a given recall value is the maximum of the precision values where the recall is higher than the given value. For the COCO dataset, mean average precision (mAP) is used. In the case of mAP, AP is averaged over a number of IoU values. AP using IoUs from 0.5 to 0.95 in steps of 0.05 is averaged for the COCO dataset.

3.1.3 Choosing YOLOv3

In terms of accuracy and speed, YOLOv3 is an excellent detection algorithm. This is best shown by direct mAP comparison on the COCO dataset. Tables 3.2 and 3.1 and Figure 3.1 display this.

YOLOv3 has a stellar performance in terms of mAP accuracy versus speed on the COCO dataset, which of course reflects the efficiency requirement emphasized by the demands of a low power system. This effect is due to specific structural and algorithmic design decisions for the method. Rather than operating a complex pipeline for detection, or applying CNN passes

Table 3.1: Older YOLO methods achieve state-of-the-art accuracy on the VOC 2007+2012 datasets while maintaining efficiency. The fastest speed and highest mAP for an IoU of 50 are highlighted in red. For networks with implementations operating at different resolutions, multiple results are provided with the input resolution given.

Method (input resolution)	VOC mAP-50	fps
YOLO [42]	63.4	45
Fast YOLO	52.7	155
YOLOv2 (288) [99]	69.0	91
YOLOv2 (352)	73.7	81
YOLOv2 (416)	76.8	67
YOLOv2 (480)	77.8	59
YOLOv2 (544)	78.6	40
Fast R-CNN [48]	70.0	0.5
Faster R-CNN [49]	76.4	5
SSD300 [41]	74.3	46
SSD500	76.8	19

Table 3.2: YOLOv3 achieves state-of-the-art accuracy on the COCO dataset while maintaining efficiency. The fastest speed and highest mAP for an IoU of 50 are highlighted in red. For networks with implementations operating at different resolutions, multiple results are provided with the input resolution given.

Method (input resolution)	COCO mAP-50	Time (ms)
YOLOv3 (320) [9]	51.5	22
YOLOv3 (416)	55.3	29
YOLOv3 (608)	57.9	51
YOLOv2 [99]	44.0	N/A
RetinaNet-50 (400) [45]	47.8	64
RetinaNet-50 (500)	50.9	72
RetinaNet-50 (600)	53.2	98
RetinaNet-50 (700)	54.2	121
RetinaNet-50 (800)	55.0	153
RetinaNet-101 (400)	49.5	81
RetinaNet-101 (500)	53.1	90
RetinaNet-101 (600)	55.2	122
RetinaNet-101 (700)	56.6	154
RetinaNet-101 (800)	57.5	198
Fast R-CNN [48]	39.9	N/A
Faster R-CNN [49]	45.3	N/A
Mask R-CNN [50]	60.0	N/A
SSD321 [41]	45.4	61
SSD513	50.4	125
DSSD321 [43]	46.1	85
DSSD513	53.3	156
ION [44]	43.2	N/A
FPN FRCN [47]	59.1	172
R-FCN [46]	51.9	85

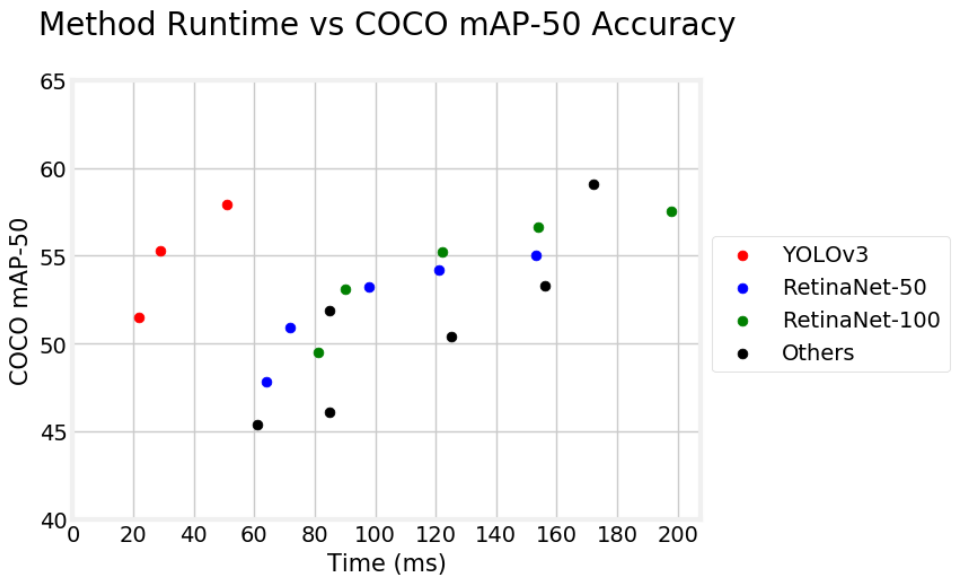


Figure 3.1: In terms of speed, YOLOv3 is a class ahead. The 'Other' methods listed are taken from Table 3.2. The trends in speed and runtime for YOLO and RetinaNet are caused by varying network input size.

for every possible bounding box region, YOLO operates by a single pass of a convolutional network from entire image to predictions.

Additionally, there are other important reasons YOLOv3 is chosen ahead of other detection algorithms. First, as of YOLOv2 the algorithm follows a training method which varies the input image resolution across different batches. This means that without retraining, the same network is able to operate at a higher fps with less accuracy, or slower speeds at state-of-the-art accuracy. The YOLO backbone network views the entire image during training, allowing it to encode contextual and appearance information about objects. This is different from region proposal methods, which limit the classifier view to a specified region. Because of this advantage, YOLO is less likely to predict false positives in background areas. This feature is important for the WIND project, as any false positives will consume resources in the object tracking phase. Additionally, looking at the full context provides generality to the algorithm, allowing higher accuracy on unexpected inputs. YOLO and many other methods have a confidence prediction, multiclass prediction, and a base convolutional neural network trained on ImageNet, all of which are helpful in real use cases. Finally, a

Python implementation that integrates well with the project while maintaining full speed is readily available.

The backbone of YOLOv3 is the darknet-53 CNN, a blend of the VGG inspired YOLOv2 darknet-19 network with residual networks and added shortcut connections. This network is described in Figure 3.2.

YOLO operates by dividing the image into an S x S grid of cells and estimating B bounding boxes [42]. Rather than a fully connected final layer which predicts the bounding boxes, a convolutional layer which predicts offsets from cluster centers of boxes of specific sizes is used. These boxes are determined by applying k-means clustering to the ground truth bounding boxes of detection datasets. Figure 3.3 shows a generalized example prediction. The CNN predicts t_x, t_y, t_w, and t_h, which relate to the center coordinates and shape of the final bounding box b as $b_x = \sigma(t_x) + c_x$, $b_y = \sigma(t_y) + c_y$, $b_w = p_w e^{t_w}$, and $b_h = p_h e^{t_h}$. p_w and p_h are the priors for the bounding box, and c_x and c_y denote the cell offset. The x and y equations predict the center coordinates of the box relative to the location of filter application using a sigmoid function. Confidence, a multiclass classification, and objectness is also predicted for each box.

YOLO is first pretrained on ImageNet for classification by minimizing binary cross entropy

$$-(y \log(p) + (1 - y) \log(1 - p))$$

, where y is the ground truth and p is the prediction. Then, the final softmax layer is discarded and the network is trained for detection by minimizing the sum of squared error

$$\frac{1}{N} \sum_{i=1}^{N} (\hat{t}_* - t_*)^2$$

where $\hat{t}_*$ is the ground truth vector and t_* is the predicted vector for N samples. In YOLOv2, multiscale training every 10 batches is introduced in training to resize the input, implicitly encoding various resolutions. By changing this resolution during testing, the same network can operate at various speeds without retraining [99].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.2: The backbone network for YOLOv3 is darknet-53, a classification network (from Redmon, 2018 [9]).

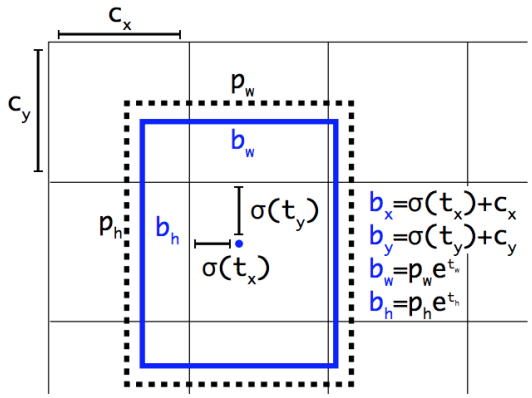


Figure 3.3: YOLOv3 predicts x and y offset from the location of filter application and w and h scaling from priors of set size.

For this nuclear source tracking framework, YOLOv3 integrates into the visual object tracking algorithm. Rather than running continuously on every new frame, YOLOv3 operates when triggered by human input. On startup, and if it is determined that enough variation has occurred in the object field to require re-initialization, the agent should trigger detection. For each object in the frame, a bounding box, class label, and confidence is returned. Given that the confidence is above a set threshold, a tracker thread is initialized corresponding to each object.

3.2 Object Tracking

This section provides the same content as section 3.1, but for object tracking rather than object detection. Appropriate datasets and metrics for evaluation of future methods are given. A case is made for SiamFC, and the details of the algorithm are described. The results of applying SiamFC to a challenge dataset are displayed in section 3.3.3.

3.2.1 Datasets

Despite the difficulties with obtaining and notating a video dataset, there are three standard public benchmarks for object tracking: the Object Tracking Benchmark (OTB) [100], Visual Object Tracking (VOT) [8], and the Amsterdam Library of Ordinary Videos for Tracking (ALOV++) [101]. OTB consists of 100 single camera, single object sequences. Of these

100 objects, 36 are human body and 26 are face videos. Each sequence is fully annotated with bounding boxes and 11 challenge attributes such as illumination, occlusion, and motion blur. This helps to distinguish the strengths and weaknesses of various trackers. VOT consists of the most updated challenge dataset from the VOT challenge, which has run annually since 2013. This dataset addresses short term, single camera, single target, model free trackers. The most recent 2017 dataset has 60 sequences, but 390 sequences have been used over the course of all challenges. Each year, the easiest sequences are replaced with more difficult videos. Every frame of these sequences is annotated with occlusion, motion blur, size change, camera motion, and illumination change information. ALOV++ consists of 89364 total frames of video, mostly sourced from real-life YouTube videos. These are chosen to cover diverse circumstances, and so contain 64 different categories of targets. The average sequence length is 9.2 seconds, with a rectangular bounding box every 5th frame, except in a few rare case of rapid movement; linear interpolation is used in between.

3.2.2 Metrics

Tracking algorithms are evaluated by metrics representing accuracy, speed, and robustness. The area of overlap between the ground truth bounding box A_G and a tracker's estimated bounding box A_T measures tracking accuracy. For a given frame, the accuracy ϕ is calculated as

$$\phi = \frac{A_G \cap A_T}{A_G \cup A_T}$$

VOT uses another metric to balance datasets of varying lengths: expected average overlap (EAO). This metric represents the expected accuracy on another dataset with similar properties. Given a large dataset of sequences with s frames, the average overlap curve $\hat{\phi}_s$ across all frames of all sequences can be calculated by

$$\hat{\phi}_s = \frac{1}{s} \sum_{i=1:s} \phi_i$$

However, the majority distribution of applicable sequences may be within a certain range lo to hi, rather than of length s. To calculate the expected average overlap value $\hat{\phi}_{lo:hi}$ on this

new range of sequences, a new averaging equation is leveraged against the expected overlap curves as

$$\hat{\phi}_{lo:hi} = \frac{1}{hi - lo} \sum_{s=lo:hi} \hat{\phi}_s$$

In order to determine the practicality of object trackers, speed is used. For the VOT challenge, equivalent filter operations (EFOs) are used to equalize tracker speed across different hardwares. Additionally, any bias from optimizations to specific hardwares is removed. EFOs are the ratio of speed on the evaluation dataset to the speed of a predefined filter operation, as shown in Figure 3.4. Tracker failures occur when the estimated bounding box drifts far enough from the target or suddenly fails repeatedly to include the target. In real use cases, re-initialization while maintaining consistency is difficult, so avoiding failures is highly important. Robustness is measured by counting the total number of failures, averaged over the size of the dataset. In the case of the VOT dataset, the tracker is re-initialized five frames after a failure.

3.2.3 Choosing SiamFC

SiamFC is the winner of the VOT 2017 Real Time Challenge, as shown in Table 3.3. Trackers for this competition must predict the bounding box for each sequence frame at a higher frequency than the video frame rate. CSRDCF++ achieves a slightly higher performance, but because it was designed in part by the challenge organizers, is not eligible for competition. Additionally, SiamFC performs well across the EAO, accuracy, and robustness metrics used

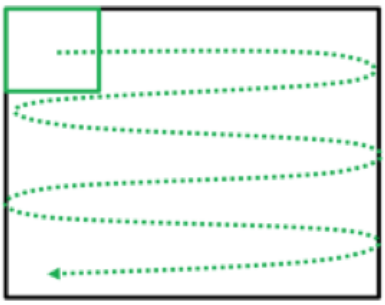


Figure 3.4: In order to calculate equivalent filter operations, a baseline for the hardware is calculated by applying a max operation on each 30x30 window of a 600x600 image (from [8]).

for the 2017 competition, as shown in Table 3.4. In order to reduce performance variation, each tracker runs a sequence 15 times. Since EFO is replaced by the real time portion in the 2017 competition, EFO results for comparable algorithms in the 2016 competition are shown in Table 3.5 and Figure 3.5.

In addition to the impressive real time performance, SiamFC also realizes other advantages over existing algorithms. Rather than using online learning to create an object model, SiamFC trains on extensive detection datasets prior to evaluation, increasing speed. This pretraining is performed currently using the new ImageNet tracking dataset, which means the learned models will share a relationship with the models learned by YOLOv3. Evaluation results also show that the model capacity has not been reached, and further training on new datasets will increase the accuracy. This method is CNN based, and so fits with current method trends in the visual object tracking research sphere; as hardware and software improvements are adapted to this trend, so the speed of SiamFC will increase. Finally, SiamFC works across multiple hardwares and systems, with an open sourced python version that works on its own, compared to many other VOT-submitted trackers which only work with the challenge dataset.

Rather than pretraining a CNN on another dataset in order to learn feature maps, and then online training the network to identify the object, SiamFC proposes to compare an exemplar image z to a candidate image x via a function $f(z, x)$ which scores higher when the

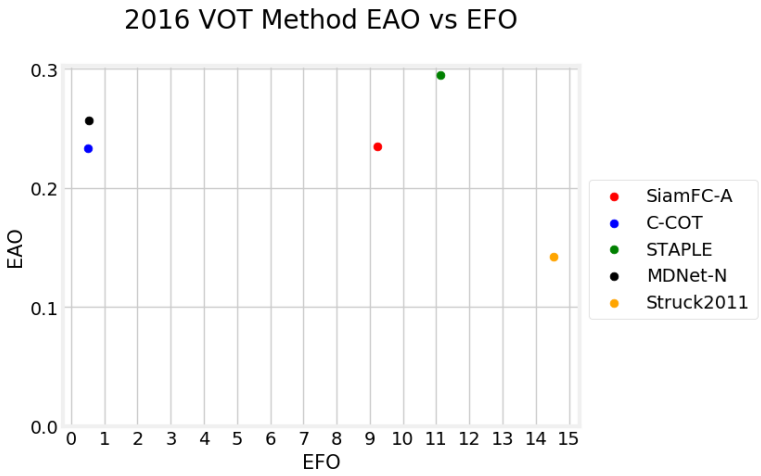


Figure 3.5: SiamFC strikes a balance between accuracy and speed. These are the VOT 2016 Challenge results from Table 3.5.

Table 3.3: SiamFC is the winner of the VOT real time 2017 challenge. The results shown are the top five entries to the real time portion only. The highest EAO is highlighted in red.

Method	EAO	A	R
SiamFC [10]	0.182	0.502	0.604
CSRDCF++ [82]	0.212	0.459	0.398
ECOhc [102]	0.177	0.494	0.571
Staple [96]	0.170	0.530	0.688
KFebT [103]	0.169	0.451	0.684

Table 3.4: SiamFC achieves state-of-the-art performance on the standard VOT 2017 challenge. Various methods from the top five 2017 methods, the real time competition, and dual 2016-2017 competition entries are shown. The highest EAO is highlighted in red.

Method	EAO	A	R
SiamFC [10]	0.188	0.502	0.585
CSRDCF++ [82]	0.229	0.453	0.370
ECOhc [102]	0.238	0.494	0.435
LSART [95]	0.323	0.493	0.218
CFWCR [8]	0.303	0.484	0.267
CFCF [104]	0.286	0.509	0.281
CCOT [79]	0.267	0.494	0.318
STAPLE [96]	0.169	0.530	0.688
GMDNetN [91]	0.157	0.513	0.696
Struck2011 [58]	0.097	0.418	1.297

Table 3.5: SiamFC achieves state-of-the-art performance as a balance between EAO and speed. These entries all competed in some form in both the 2016 and 2017 VOT challenges. All results below are obtained on the 2016 dataset.

Method	EAO	EFO
SiamFC-A [10]	0.235	9.213
C-COT [79]	0.331	0.507
STAPLE [96]	0.295	11.114
MDNet-N [91]	0.257	0.534
Struck2011 [58]	0.142	14.584

images containing the same object. Applying a CNN as the function f via siamese networks, which are common for similarity learning, allows the initial appearance of the object to be searched for against the remaining frames of the video. Siamese networks implement the function f by applying an identical transformation φ to both inputs, and then combining the representations by a similarity metric g as

$$f(z, x) = g(\varphi(z), \varphi(x))$$

This function is further improved by using a fully-convolutional network as the embedding φ and cross correlation $*$ as the similarity metric as

$$f(z, x) = \varphi(z) * \varphi(x) + b\mathbb{1}$$

where $b\mathbb{1}$ is a region of value b. This causes the candidate image to be applied on a dense grid of translated sub-windows in a single evaluation. SiamFC uses a search region four times the previous size, penalizing displacements from the center with a cosine window. The entire network, with specific details given in Figure 3.6, produces a final score map as shown in Figure 3.7.

				Activation size		
Layer	Support	Chan. map	Stride	for exemplar	for search	chans.
				127×127	255×255	$\times 3$
conv1	11×11	96×3	2	59×59	123×123	$\times 96$
pool1	3×3		2	29×29	61×61	$\times 96$
conv2	5×5	256×48	1	25×25	57×57	$\times 256$
pool2	3×3		2	12×12	28×28	$\times 256$
conv3	3×3	384×256	1	10×10	26×26	$\times 192$
conv4	3×3	384×192	1	8×8	24×24	$\times 192$
conv5	3×3	256×192	1	6×6	22×22	$\times 128$

Figure 3.6: The backbone architecture of the of the convolutional function is similar to AlexNet (from Bertinetto, 2016 [10]).

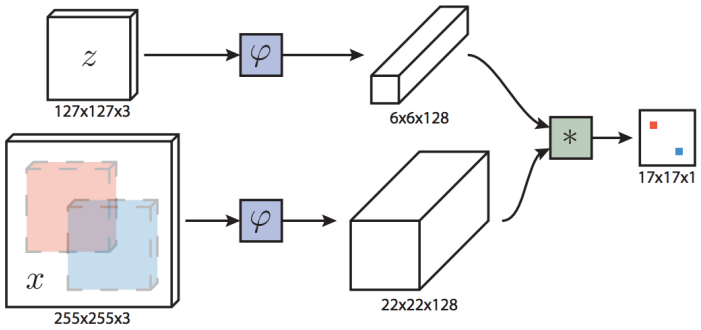


Figure 3.7: A score map is output for all translated sub-windows in the search image (from Bertinetto, 2016 [10]).

The CNN θ is trained by applying stochastic gradient descent to

$$arg\ min_{\theta}\mathbb{E}_{(z,x,y)}L(y, f(z, x : \theta))$$

where $L(y, v)$ is the loss of the score map. It is defined as

$$L(y, v) = \frac{1}{|D|}\sum_{u \in D} l(y[u], v[u])$$

where u is a position in the map D, and $l(y, v)$ is the logistic loss for a position. It is defined as

$$l(y, v) = \log(1 + e^{-yv})$$

where y is the ground truth label and v is the real valued score of one image pair.

3.3 Results

3.3.1 Data

Multiple datasets are used for performance evaluation of the detection and tracking methods. On April 15, 2013 during the Boston Marathon, two bombs were set off resulting in 3 fatalities and 176 injuries. In the aftermath of the explosions, CCTV footage of the crowded scenes shows two suspects [105]. The Boston bombing dataset consists of six video sequences from Mashable during which one or two of the suspects navigates a crowd. Ground truth

highlighting is performed directly on the video using a black and white circle, but no official bounding box information is provided, so comparisons in performance must be done by hand.

In order to capture the differences in performance across computational platforms of various capability, testing is performed on two systems. The first is a Mid 2015 MacBook Pro Retina, with four 2.2 GHz Intel Core i7s and no graphics card use. The second is a Linux desktop with eight 3.4 GHz Intel Core i7s and a GeForce GTX 1070.

3.3.2 Object Detection

YOLOv3 performs at different accuracies with different network inputs. Using a smaller input resolution is more efficient and is able to run at a higher fps, but provides less accuracy. Using the Boston sequences, the difference in accuracy measured by mAP can be observed more clearly. Figures 3.8, 3.9, 3.10, 3.11, 3.12, and 3.13 provide visual comparisons, as well as examples of the detection capabilities. All objects must be above 25% confidence to be detected.

YOLOv3 operates at an increased efficiency when run on a GPU. The difference in fps between using the tiny and normal network is more pronounced on the CPU (10x) compared to the GPU (2x), but the difference between CPU and GPU performance is 50x. With a GPU, detection occurs faster than the framerate, allowing real time detection. Table 3.6 shows these results across the two architectures investigated.

Table 3.6: The difference in fps between GPU and CPU architectures for YOLOv3 is large. The fps values from running on each Boston bombing sequence are shown.

Architecture	Network	Boston 1	Boston 2	Boston 3	Boston 4	Boston 5	Boston 6
GPU	Tiny	52.8 fps	50.8 fps	55.0 fps	53.8 fps	52.4 fps	51.6 fps
GPU	Normal	22.0 fps	22.0 fps	22.2 fps	21.3 fps	21.5 fps	21.9 fps
CPU	Tiny	1.4 fps	1.4 fps	1.4 fps	1.4 fps	1.4 fps	1.2 fps
CPU	Normal	0.13 fps	0.13 fps	0.13 fps	0.13 fps	0.13 fps	0.14 fps



(a) (b) (c) (d)

(e) (f) (g) (h)

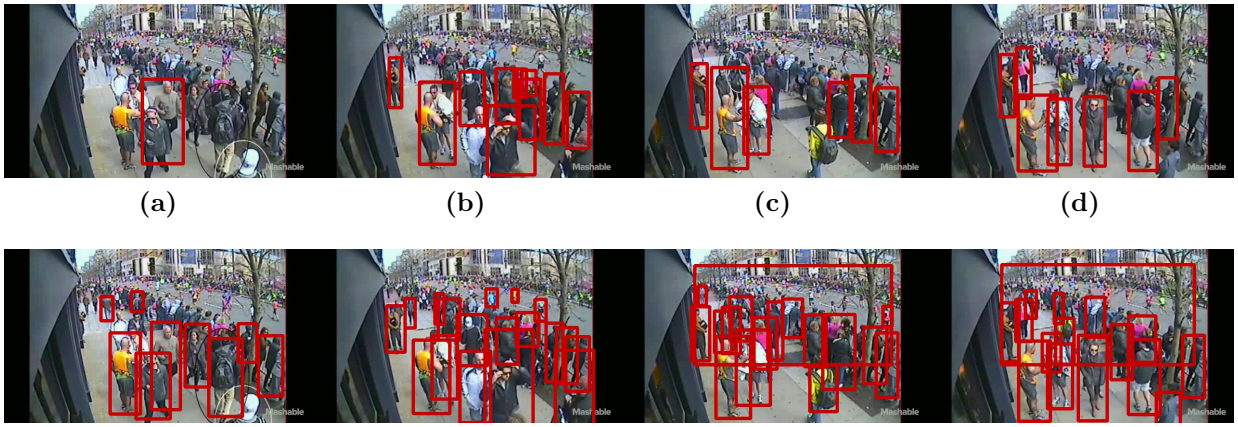
Figure 3.8: Running the YOLOv3 tiny and normal networks on Boston sequence 1 results in a difference in detection capabilities. (tiny a-d, normal e-h)



(a) (b) (c) (d)

(e) (f) (g) (h)

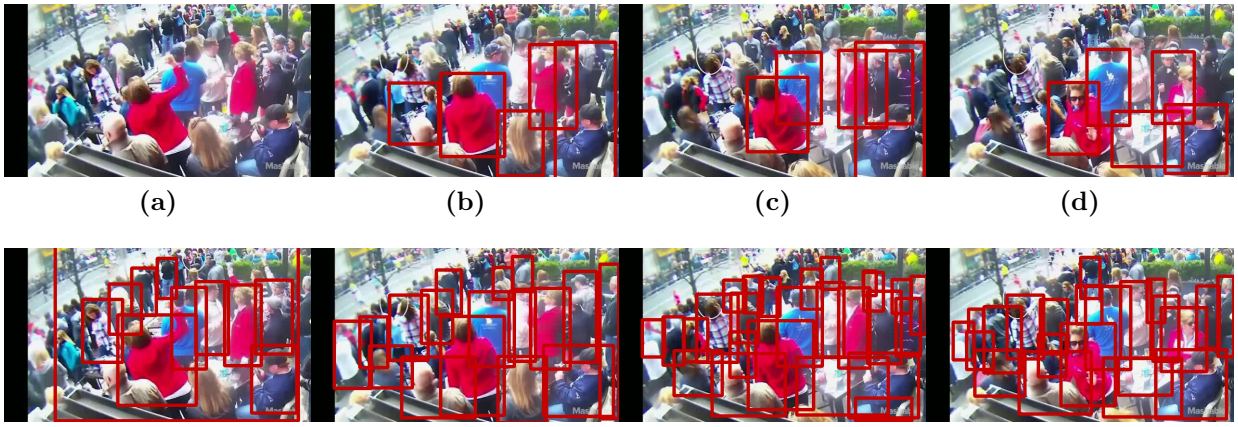
Figure 3.9: In crowds, the tiny network misses many partially occluded or rotated individuals. (tiny a-d, normal e-h)



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.10: One artifact of the network at a low confidence threshold is the tendency to mislabel the entire crowd as one person. (tiny a-d, normal e-h)



(a) (b) (c) (d)

(e) (f) (g) (h)

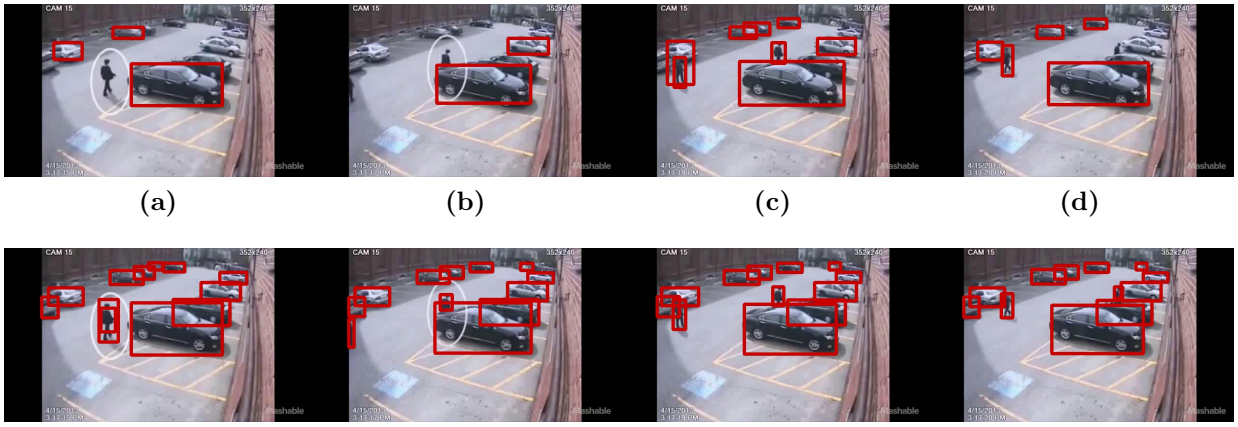
Figure 3.11: The normal network can capture many individuals in a crowd, but is overwhelmed by crowds appearing farther back in frame. (tiny a-d, normal e-h)



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.12: Sequences like this, with a single individual, give the normal network no challenge. (tiny a-d, normal e-h)



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.13: On sequence six, the tiny network has trouble detecting the target, and misses the majority of the parked cars (tiny a-d, normal e-h)

3.3.3 Object Tracking Results

.

SiamFC is not the most accurate object tracking method, but performs well on the Boston bombing dataset. The results of tracking the targets given initializations using ground truth are shown below for each sequence in Figures 3.14, 3.15, 3.16, 3.17, 3.18, and 3.19.

Although SiamFC can run at one frame per second on a CPU as in Table 3.7, in real time the movement possible in one second disrupts the tracking capabilities. However, using a GPU allows the network to track in real time, at a framerate below the input framerate. Using a more powerful GPU would further increase the framerate.

3.3.4 Multi-Object Tracking

SiamFC and YOLOv3 work well separately, but also operate together to create a functional multi-object tracker. Figures 3.20, 3.21, 3.22, 3.23, and 3.24 show the results of first detecting the objects in frame, and then tracking the subjects based on those initial detection bounding boxes. The Boston bombing sequences are shown except sequence two, on which the tracking failed across multiple initializations.

Table 3.7: SiamFC tracks at 10 times the frames per second on a GPU compared to a CPU.

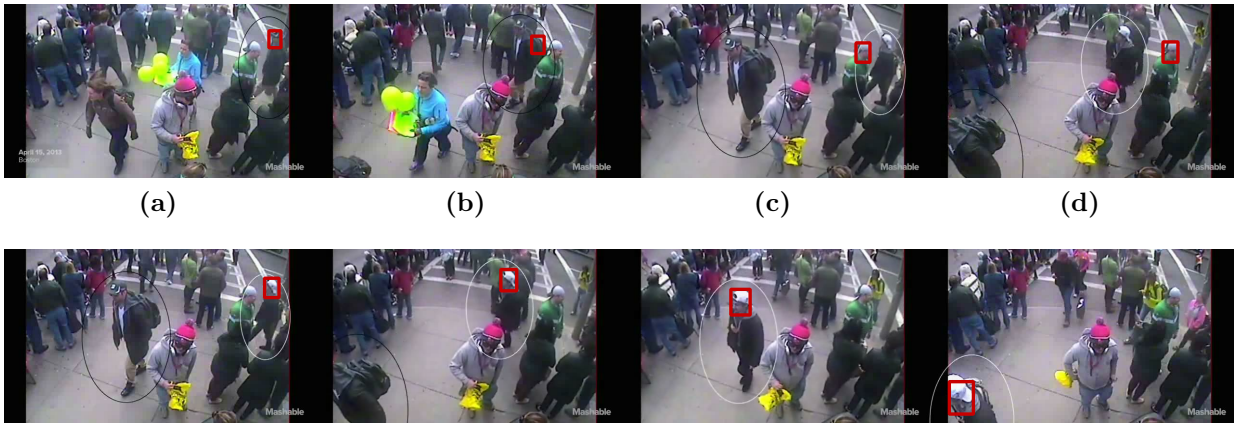
Architecture	Boston 1	Boston 2	Boston 3	Boston 4	Boston 5	Boston 6
GPU	10.3 fps	9.9 fps	10.9 fps	11.1 fps	10.4 fps	10.4 fps
CPU	1.4 fps	1.4 fps	1.4 fps	1.3 fps	1.2 fps	1.3 fps



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.14: Both subjects are successfully tracked.



(a) (b) (c) (d)

(e) (f) (g) (h)

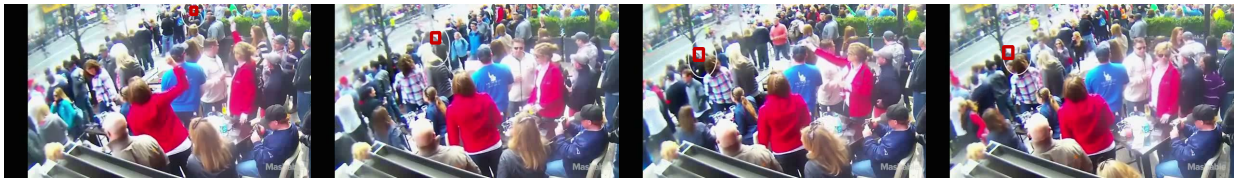
Figure 3.15: One of the subjects is lost due to occlusion.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.16: Here, the backpack rotation and crowd occlusion cause a loss of one subject.



(a) (b) (c) (d)

Figure 3.17: Provided with a clear, unoccluded face, the network can achieve high accuracy with little information.



(a) (b) (c) (d)

Figure 3.18: Even though the subject rotates, tracking continues.



(a) (b) (c) (d)

Figure 3.19: Despite the low resolution and partial occlusion, SiamFC tracks the subject into the car.



(a) (b) (c) (d)

Figure 3.20: Both targets are detected and then tracked for the duration of the sequence.



(a) (b) (c) (d)

Figure 3.21: Both targets are detected, but only one is tracked for the duration of the sequence.



(a) (b) (c) (d)

Figure 3.22: As the targets walks through the crowd, he comes into view, is detected, and then is tracked until his body is occluded again.



(a) (b) (c) (d)

Figure 3.23: Even when initialized by the detector, the tracker can still remain robust to the target's rotation.



(a) (b) (c) (d)

Figure 3.24: Although the target is lost to the detector over the course of the sequence as seen above, the tracker is able to continue to localize given the detector's initial bounding box.

Chapter 4

Active Tracking System

In this chapter, information and results related to the proof-of-concept algorithm and method fusion are presented. A general overview is given with emphasis on how each algorithm integrates into the overall method. Finally, details about and results of visual and nuclear algorithm fusion are presented.

4.1 System Overview

In this thesis, a proof-of-concept algorithm is designed to track a nuclear source by combining radiosotope direction estimation and visual object tracking algorithms. The algorithm was designed for use in a wearable backpack, with the goal of continually assisting a human agent. This goal is achieved in the form of a modular structure that both serves as a starting point for combining various areas of research, and successfully provides assistance by predicting which targets in the current field of view are most likely to be carrying a nuclear source. This framework is also ready for updates as future improvements and additions are contributed. Figure 4.1 shows the overall flow path of the system.

The current system is implemented from scratch in Python 3.6 and C. The individual vision algorithms are written in C languages in order to optimize their speeds, while Python is used to gracefully control inputs, data, and the main program flow. YAML is used for configuration setup, and OpenCV controls the various camera and video functions. The database runs in straightforward Python without dependencies. This software works on

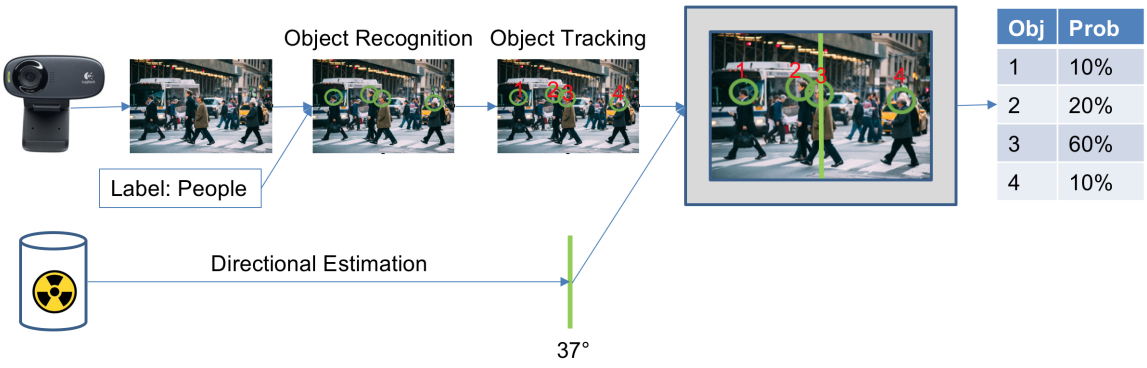


Figure 4.1: An overview of the system flow path.

MacOS, Windows, and Linux systems. An advanced graphics card is required for the system to function accurately and at full speed, although various components may be run individually on a CPU.

In order to prevent rapid obsolescence of this prototype during the course of the WIND project, adaptability, modularity, and convenience are prioritized. Python provides a strong base for these three goals. As Python has a library for almost anything, it is likely that new smart device integrations or networking protocols will already be handled by existing open-source codes. Although currently the algorithms are implemented in C, Python is also easily adaptable to other languages if future wrappers are needed. This modularity is also reflected in the directory structure created and code written, so that additions and deletions are safe and convenient to execute without extensive repository knowledge. Finally, the initial algorithms chosen are adaptable to various machine capabilities, and can be adjusted through the system configuration. This allows the same, slightly less accurate tracking to occur on less powerful computers.

Although the final system use case involves an active agent, the current control scheme is more reflective of the laboratory environment. In the field, the system assumes a nearby source of known isotope. From this starting point, the agent can observe the system output to determine in which direction to travel based on the nuclear detector prediction. As she travels, the agent triggers the visual systems to operate, which automatically detect and track possible targets on screen, making a prediction of the most likely carrier. As the environment changes due to crowd or traffic movement relative to the agent, she repeats this

process while drawing nearer to the source, attempting to keep the most likely carrier and source direction in front of her. When these data agree, it is up to the agent to address the found source. In the lab, individual algorithms and system components can be trained and evaluated separately using terminal or IDE control of the program via an extensive list of functions and options.

4.1.1 Nuclear Directionality

The wearable detector setup described in Chapter 2 provides a predicted angle of the source relative to the detector, along with a corresponding estimate of the error, all in the form of an angular probability distribution. This information is extracted from detector count rate data using gradient boosting regression regardless of the detector setup. The direction estimation algorithm must be trained for any given detection setup, but the design remains the same beyond the number of input features. This part of the framework is modular, so that as more accurate methods or more complexity such as isotope identification is added, the changes are confined to a small part of the program.

4.1.2 Visual Object Tracking

The vision tracking system described in Chapter 3 provides estimates in two complementary stages. First, an advanced object detection tool works in real time to identify important objects within the trigger frame, such as vehicles, bags, and people. Currently YOLOv3 performs this object detection. Although YOLOv3 does not perform with the highest accuracy on the standard Common Objects in Context (COCO) dataset, it does trade-off with speed to be the best performer in real time. Additionally, the single neural network is adjustable to scale accuracy and efficiency to the needs of the system. At the present time, as multiple hardware specifications are used, this functionality is desirable. The individual results of detection are passed as the initialization bounding boxes and labels to a tracking algorithm. Next, an object tracker continually returns the position of each object found by detection. Currently SiamFC performs this object tracking. SiamFC operates with stellar

accuracy and robustness on the Visual Object Tracking (VOT) challenge dataset. Both of these methods are modular, and are only connected via their inputs and outputs.

4.2 Sequential Bayesian Inference

In order to provide the final visual and probabilistic output, sequential Bayesian inference is used to fuse the source directionality estimate and tracked objects' positions. Bayesian inference works using the simple Bayesian equation given in section 1.2.2, but is discussed further here. Given a field of N objects, consider the event E_i that a certain object i is carrying the source with a certain probability $P(E_i)$. Let D be the data obtained from the algorithms, such that $P(E_i|D)$ is the posterior probability of each object guess given the data. If each object guess hypothesis is assumed to be true, the likelihood $P(D|E_i)$ for a given angle is retrieved. These pieces are put together using Bayes' theorem

$$P(E_i|D) = \frac{P(D|E_i)P(E_i)}{P(D)}$$

where $P(D)$ is computed using the law of total probability as

$$P(D) = \sum_{i=1}^{N} P(D|E_i)P(E_i)$$

This entire process is not static, but instead iterates dynamically for each data input step. In the first case, equal priors $P(E_i) = 1/N$ are assumed. The posterior probabilities are calculated for each object, and then the new priors are set equal to the old posteriors $P(E_i) \coloneqq P(E_i|D)$ where $\coloneqq$ is the assignment operator. This process continues until a reset from equal priors occurs, or until the algorithm is halted.

Here is a detailed description of the process over two iterations which match the information in Table 4.1. Consider three people moving around the detector, where person 1 is carrying the source. Equal prior probabilities are assumed to start, and the angles of the pedestrians are given relative to the source angle, such that 0° is the source angle. In the first iteration, objects 2 and 3 are equally spaced on either side of the estimated source angle,

Table 4.1: This table shows two iterations of Bayesian updating for three objects. Starting with equal priors, the angle relative to the source is given for each object at each step, and the corresponding posteriors are calculated as above using Bayes' theorem.

i	$P(E_i)$	D_1	$P(D_1\|E_i)$	$P(D_1)$	$P(E_i\|D_1)$	D_2	$P(D_2\|E_i)$	$P(D_2)$	$P(E_i\|D_2)$
1	0.33	0°	0.7	0.33	0.7	0°	0.7	0.55	0.89
2	0.33	−45°	0.15	0.33	0.15	−90°	0.05	0.55	0.01
3	0.33	45°	0.15	0.33	0.15	20°	0.35	0.55	0.1

resulting in equal likelihoods and posterior probabilities $P(E_i|D_i)$. These prior probabilities are then assigned as the priors during the second iteration. At this point, the pedestrians have traveled rotationally relative to the source, resulting in three different likelihoods. The final posterior probabilities are now an aggregate of the prior probabilities and both sets of likelihoods.

4.3 Results

Over time, this fusion method becomes more confident about the source carrier. This occurs even under challenging conditions, with low accuracy from the directional estimator. Table 4.2 shows the number of iterations the algorithm takes to reach 80% confidence. These results assume five possible carriers are present: one holding the source, and four standing all together at various degrees of separation from the source. All carriers are standing still throughout. With a high accuracy of 10 degrees, the confidence quickly converges. More important, however, is the result at a low accuracy of 50 degrees. In this case, it is possible to eliminate possible carriers just 45 degrees from the source after only six iterations.

Table 4.2: The fusion algorithm takes multiple iterations to reach 80% confidence when five possible carriers are present at a given angular split from each other.

		RMSE		
		10°	25°	50°
Split	10°	6	43	60
	30°	1	5	14
	45°	1	1	6

A few simulations and challenge experiments are presented to evaluate the effectiveness of this fusion method. In the first three, a Gaussian distribution as in Figure 4.2 is used to calculate the likelihood values for each source as the sequence runs, but the results are not retained across frames. In the final example, the posterior values from the previous frame are used as the priors for the next frame.

The first is an orbit simulation, shown in Figure 4.3, where all of the angle and position data are set programatically at startup. The red planet carries the source around the small black detector. Over the course of the ten second video, the system becomes more confident. In the next two examples the individuals are tracked manually and a fabricated source is placed on one individual's center of mass. Four sequential screenshots are shown from each clip sequence, taken from music videos by Beyonce and OK Go. Sequence one, "White Knuckles", is shown in Figure 4.4, while sequence two, "All the Single Ladies", is shown in Figure 4.5. For both of these results the visual display of angle is approximated manually. The final simulation, shown in Figure 4.6, uses the real tracking results from YOLOv3 and SiamFC. The confidence values are presented over the head of each possible carrier; the results are calculated as if suspect with a white outline has the source. The directional estimation capability is assumed to be a Gaussian distribution of variance 50 degrees.

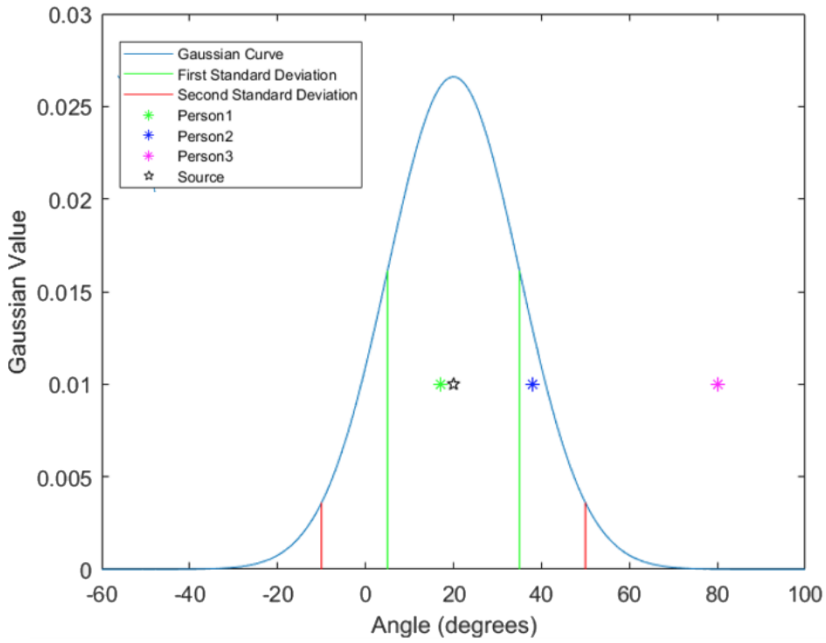
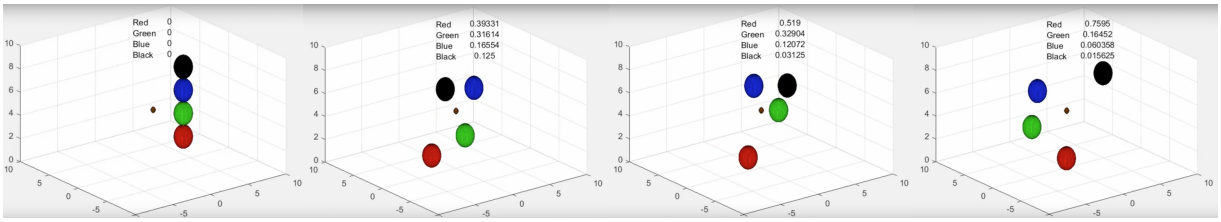
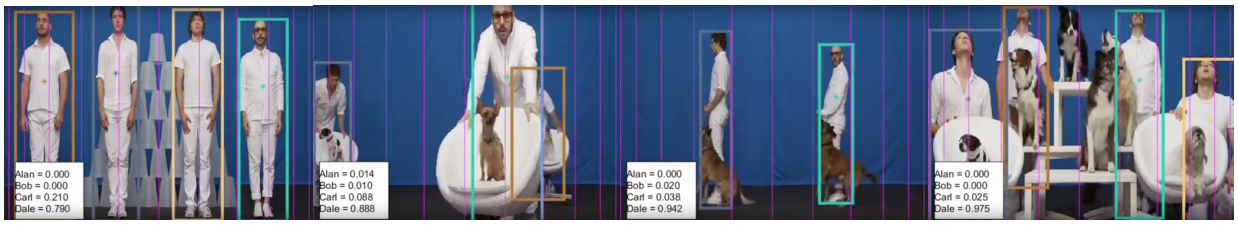


Figure 4.2: A Gaussian distribution is used to calculate likelihood for the planets, OK Go, and Beyonce sequences.



(a) **(b)** **(c)** **(d)**

Figure 4.3: Four frames from a MATLAB simulation of orbiting bodies around a detector. The red body has the source, and over time the system predictions increase in accuracy. Video created by Steven Patrick.



(a) **(b)** **(c)** **(d)**

Figure 4.4: Four frames from Ok Go's music video "White Knuckles" [11] are shown. Dale, outlined in blue, is holding a fabricated source at his center of mass. The tracking information is input manually. Video created by Steven Patrick.



(a) **(b)** **(c)** **(d)**

Figure 4.5: Four frames from Beyonce's music video "All the Single Ladies" [12] are shown. Beyonce is holding a fabricated source at her center of mass. The tracking information is input manually. Video created by Steven Patrick.



(a) **(b)** **(c)** **(d)**

Figure 4.6: Four frames from the Boston marathon bombing video dataset are shown. The suspect outlined in a white circle is holding a fabricated source at his center of mass. The tracking information is calculated using YOLOv3 and SiamFC.

Chapter 5

Conclusions and Future Work

Previously, most localization methods assumed a stationary or moving detector setup with the goal of accurately identifying the coordinates of a stationary source. This thesis takes a large intermediary step towards localizing a moving source, assisted by advanced object tracking techniques. The proposed algorithm can estimate carrier likelihood for objects in its field of view, and is designed to assist a pedestrian agent wearing the backpack detector built by the IDEAS for WIND research group. This setup takes advantage of recent advances in detector, camera, and computer technologies to meet the challenging physical limitations.

This thesis made three contributions. First, a generalized directional estimator was proposed. Second, two state-of-the-art visual object detection and visual object tracking methods were combined into a single tracking algorithm. Third, those outputs were fused to produce a real time radioisotope tracking algorithm.

The directional estimator operates via gradient boosting regression to predict radioisotope direction with a variance of 50 degrees when trained on a simple laboratory dataset. Under conditions similar to other state-of-the-art methods, the accuracy is comparable, at an RMSE of 10 degrees. YOLOv3 and SiamFC are chosen by evaluating advanced visual tracking methods in terms of speed and efficiency across multiple architectures, and in terms of accuracy on datasets like the Visual Object Tracking (VOT) Challenge and Common Objects in Context (COCO). YOLOv3 can process images at 20 FPS on a GTX 1070 GPU while achieving a mAP of 57.9% on the COCO dataset, and SiamFC runs at 10 FPS while performing with an EAO of 0.235 on the VOT 2016 Challenge. The resultant tracking

algorithm from combining their functions operates in real time. The outputs of direction estimation and visual tracking are fused using sequential Bayesian inference to predict carrier likelihood. Assuming a group of five possible carriers standing within 10 degrees of each other, this fusion method identifies the carrier with 80% confidence within one minute, even under challenging error conditions. Using lab trials evaluated by hand on visual and nuclear data, and a synthesized challenge dataset using visual data from the Boston Marathon attack, it can be observed that this prototype system advances the state-of-the-art towards localization of a moving source.

In order to improve the contributions further, the following improvements are suggested for further study. More data should be collected under various source isotope and relative source strength conditions to better train and evaluate the directional estimator. The regression method used for prediction should be tailored to this dataset. Because there are constant improvements to the state-of-the-art in visual algorithms and the technical capabilities of laptop computer systems, the methods chosen in this thesis will need to be updated to the new state-of-the-art in future years for optimal performance. These methods should be chosen using the resources presented. Finally, the fusion algorithm requires further evaluation on challenge data collected using both the camera and detector from the real backpack system.

Bibliography

[1] Tungsten, "Alfa beta gamma radiation," 2005. xii, 3

[2] T. Srivastava, "Getting your clustering right," 2013. xii, 9

[3] Cyc, "Svm max sep hyperplane with margin," 2008. xii, 11, 12

[4] G. Saini, "Artificial neural network," 2017. xii, 15

[5] Glosser.ca, "Colored neural network," 2013. xii, 16

[6] M. Dunn, "Fourier transforms in image processing," 2017. xii, 19

[7] F. Orakzai, "Handwritten digits recognition using deep learning," 2016. xii, 20

[8] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. C. Zajc, T. Vojr, G. Hger, A. Lukeic, A. Eldesokey, G. Fernndez, . Garca-Martn, A. Muhic, A. Petrosino, A. Memarmoghadam, A. Vedaldi, A. Manzanera, A. Tran, A. Alatan, B. Mocanu, B. Chen, C. Huang, C. Xu, C. Sun, D. Du, D. Zhang, D. Du, D. Mishra, E. Gundogdu, E. Velasco-Salido, F. S. Khan, F. Battistone, G. R. K. S. Subrahmanyam, G. Bhat, G. Huang, G. Bastos, G. Seetharaman, H. Zhang, H. Li, H. Lu, I. Drummond, J. Valmadre, J. c. Jeong, J. i. Cho, J. Y. Lee, J. Noskova, J. Zhu, J. Gao, J. Liu, J. W. Kim, J. F. Henriques, J. M. Martnez, J. Zhuang, J. Xing, J. Gao, K. Chen, K. Palaniappan, K. Lebeda, K. Gao, K. M. Kitani, L. Zhang, L. Wang, L. Yang, L. Wen, L. Bertinetto, M. Poostchi, M. Danelljan, M. Mueller, M. Zhang, M. H. Yang, N. Xie, N. Wang, O. Miksik, P. Moallem, P. V. M, P. Senna, P. H. S. Torr, Q. Wang, Q. Yu, Q. Huang, R. Martn-Nieto, R. Bowden, R. Liu, R. Tapu, S. Hadfield, S. Lyu, S. Golodetz, S. Choi, T. Zhang, T. Zaharia, V. Santopietro, W. Zou, W. Hu, W. Tao, W. Li, W. Zhou, X. Yu, X. Bian, Y. Li, Y. Xing, Y. Fan, Z. Zhu, Z. Zhang, and Z. He, "The visual object tracking vot2017 challenge results," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 1949–1972, Oct 2017. xii, xiii, 26, 27, 48, 50, 52

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. xiii, 25, 41, 44, 47

[10] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," *CoRR*, vol. abs/1606.09549, 2016. xiii, 28, 52, 53, 54

[11] O. Go, "Ok go - white knuckles - official video." xv, 69

[12] Beyonc, "Beyonc - single ladies (put a ring on it) (video version)." xv, 70

[13] G. F. Knoll, *Radiation detection and measurement / Glenn F. Knoll*. Wiley New York, 2nd ed. ed., 1989. 3, 5, 6

[14] J. Shultis and R. Faw, *Radiation Shielding*. Prentice Hall PTR, 1996. 4

[15] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. 6

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2 ed., 2001. 8, 9, 10, 15

[17] E. Alpaydin, *Introduction to Machine Learning Third Edition*. The MIT Press, 2014. 11, 12, 14, 16

[18] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson Education, 4th ed. ed., 2018. 17, 18, 19, 20

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86(11):2278-2324, November 1998. 20, 28

[20] R. C. Runkle, M. J. Myjak, S. D. Kiff, D. E. Sidor, S. J. Morris, J. S. Rohrer, K. D. Jarman, D. M. Pfund, L. C. Todd, R. S. Bowler, and C. A. Mullen, "Lynx: An unattended sensor system for detection of gamma-ray and neutron emissions from special nuclear materials," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 598, no. 3, pp. 815 – 825, 2009. 21, 22

[21] M. Kamuda, J. Stinnett, and C. J. Sullivan, "Automated isotope identification algorithm using artificial neural networks," *IEEE Transactions on Nuclear Science*, vol. 64, pp. 1858–1864, July 2017. 21, 22

[22] R. Ştefănescu, K. Schmidt, J. Hite, R. C. Smith, and J. Mattingly, "Hybrid optimization and Bayesian inference techniques for a non-smooth radiation detection problem," *International Journal for Numerical Methods in Engineering*, vol. 111, pp. 955–982, Sept. 2017. 21, 22

[23] A. A. R. Newaz, S. Jeong, H. Lee, H. Ryu, and N. Y. Chong, "Uav-based multiple source localization and contour mapping of radiation fields," *Robotics and Autonomous Systems*, vol. 85, pp. 12 – 25, 2016. 21, 22

[24] J. M. Hite, J. K. Mattingly, K. L. Schmidt, R. Stefanescu, and R. Smith, "Bayesian metropolis methods applied to sensor networks for radiation source localization," *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 389–393, 2016. 21, 22

[25] E. A. Miller, S. M. Robinson, K. K. Anderson, J. D. McCall, A. M. Prinke, J. B. Webster, and C. E. Seifert, "Adaptively reevaluated bayesian localization (arbl): A novel technique for radiological source localization," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 784, pp. 332 – 338, 2015. Symposium on Radiation Measurements and Applications 2014 (SORMA XV). 21, 22

[26] R. D. Penny, T. M. Crowley, B. M. Gardner, M. J. Mandell, Y. Guo, E. B. Haas, D. J. Knize, R. A. Kuharski, D. Ranta, R. Shyffer, S. Labov, K. Nelson, B. Seilhan, and J. D. Valentine, "Improved radiological/nuclear source localization in variable norm background: An mlem approach with segmentation data," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 784, pp. 319 – 325, 2015. Symposium on Radiation Measurements and Applications 2014 (SORMA XV). 21

[27] J. Towler, B. Krawiec, and K. Kochersberger, "Radiation mapping in post-disaster environments using an autonomous helicopter," *Remote Sensing*, vol. 4, no. 7, pp. 1995–2015, 2012. 22

[28] C. M Combes, P. Dorenbos, C. Eijk, K. W Krak, and H. U Guk, "Optical and scintillation properties of pure and ce¿-doped cs liycl and li ycl : Ce¿ crystals," vol. 82, pp. 299–305, 01 1999. 22

[29] C. Piemonte, R. Battiston, M. Boscardin, G. F. D. Betta, A. D. Guerra, N. Dinu, A. Pozza, and N. Zorzi, "Characterization of the first prototypes of silicon photomultiplier fabricated at itc-irst," *IEEE Transactions on Nuclear Science*, vol. 54, pp. 236–244, Feb 2007. 22

[30] R. Forster and T. N. K. Godfrey, "Mcnp- a general monte-carlo code for neutrons and photon transport," vol. 240, pp. 33–, 01 1985. 22

[31] M. J. King, B. Harris, M. Toolin, R. M. DuBord, V. J. Skowronski, M. A. LuSoto, R. J. Estep, S. M. Brennan, B. R. Cosofret, and K. N. Shokhirev, "An urban environment simulation framework for evaluating novel distributed radiation detection architectures," in *2010 IEEE International Conference on Technologies for Homeland Security (HST)*, pp. 446–452, Nov 2010. 22

[32] M. J. Willis, S. E. Skutnik, and H. L. Hall, "Detection and positioning of radioactive sources using a four-detector response algorithm," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 767, pp. 445 – 452, 2014. 23

[33] R. W. S. M. P. G. C. T. Sanjoy Mukhopadhyay, Richard Maurer, "Exploitation of geometric occlusion and covariance spectroscopy in a gamma sensor array," 2013. 23

[34] B. Ayaz-Maierhafer, C. G. Britt, A. J. August, H. Qi, C. E. Seifert, and J. P. Hayward, "Design optimization for a wearable, gamma-ray and neutron sensitive, detector array with directionality estimation," *Nuclear Instruments and Methods in Physics Research*

Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 870, pp. 131 – 139, 2017. 23, 29, 32, 34

[35] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 23–38, Jan 1998. 24

[36] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627–1645, Sept 2010. 24

[37] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013. 24

[38] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. 24, 28

[39] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013. 24

[40] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," *CoRR*, vol. abs/1312.2249, 2013. 24

[41] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. 24, 43, 44

[42] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. 24, 28, 43, 46

[43] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," *CoRR*, vol. abs/1701.06659, 2017. 25, 44

[44] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," *CoRR*, vol. abs/1512.04143, 2015. 25, 44

[45] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. 25, 44

[46] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. 25, 44

[47] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. 25, 44

[48] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. 25, 43, 44

[49] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. 25, 43, 44

[50] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. 25, 44

[51] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 26, 27

[52] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 26

[53] M. Godec, P. Roth, and H. Bischof, "Hough-based tracking of non-rigid objects," *Computer Vision and Image Understanding*, vol. 117, no. 10, pp. 1245 – 1256, 2013. 26

[54] M. Poostchi, H. Aliakbarpour, R. Viguier, F. Bunyak, K. Palaniappan, and G. Seetharaman, "Semantic depth map fusion for moving vehicle detection in aerial video," pp. 32–40, Jul 2016. 26

[55] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, pp. 125–141, May 2008. 26

[56] D. Chen, Z. Yuan, Y. Wu, G. Zhang, and N. Zheng, "Constructing adaptive complex cells for robust visual tracking," pp. 1113–1120, 12 2013. 26

[57] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," vol. 33, pp. 1619–1632, 08 2011. 27

[58] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels.," in *ICCV* (D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, eds.), pp. 263–270, IEEE Computer Society, 2011. 27, 52

[59] J. Gao, H. Ling, W. Hu, and J. Xing, *Transfer Learning Based Visual Tracking with Gaussian Processes Regression*, pp. 188–203. Cham: Springer International Publishing, 2014. 27

[60] A. González, R. Martín-Nieto, J. Bescós, and J. M. Martínez, "Single object long-term tracker for smart control of a ptz camera," in *Proceedings of the International Conference on Distributed Smart Cameras*, ICDSC '14, (New York, NY, USA), pp. 39:1–39:6, ACM, 2014. 27

[61] J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi, "Visual tracking using attention-modulated disintegration and integration," June 2016. 27

[62] T. Vojir, J. Matas, and J. Noskova, "Online adaptive hidden markov model for multi-tracker fusion," *CoRR*, vol. abs/1504.06103, 2015. 27

[63] M. Maresca and A. Petrosino, "Matrioska: A multi-level approach to fast tracking by learning," in *Image Analysis and Processing ICIAP 2013*, vol. 8157 of *Lecture Notes in Computer Science*, pp. 419–428, Springer Berlin Heidelberg, 2013. 27

[64] G. Nebehay and R. P. Pflugfelder, "Clustering of static-adaptive correspondences for deformable object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 2784–2791, 2015. 27

[65] T. Vojíř and J. Matas, *The Enhanced Flock of Trackers*, pp. 113–136. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. 27

[66] M. E. Maresca and A. Petrosino, *Clustering Local Motion Estimates for Robust and Efficient Object Tracking*, pp. 244–253. Cham: Springer International Publishing, 2015. 27

[67] T. Vojíř and J. Matas, "Robustifying the flock of trackers," in *CVWW '11: Proceedings of the 16th Computer Vision Winter Workshop*, (Inffeldgasse 16/II, Graz, Austria), pp. 91–97, Graz University of Technology, February 2011. 27

[68] T. Vojir, J. Noskova, and J. Matas, *Robust Scale-Adaptive Mean-Shift for Tracking*, pp. 652–663. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. 27

[69] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," 2017. 27

[70] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. ehovin, T. Vojr, G. Hger, A. Lukei, G. Fernandez Dominguez, A. Gupta, A. Petrosino, A. Memarmoghadam, A. Garcia-Martin, A. Sols Montero, A. Vedaldi, A. Robinson, A. Ma, A. Varfolomieiev, and Z. Chi, "The visual object tracking vot2016 challenge results," pp. 777–823, 10 2016. 27

[71] M. Danelljan, G. Hger, F. Khan, and Felsberg, "Learning spatially regularized correlation filters for visual tracking," *International Conference on Computer Vision*, 2015. 28

[72] E. Gundogdu and A. Alatan, "Spatial windowing for correlation filter based visual tracking," *ICIP*, 2016. 28

[73] M. Danelljan, G. Hger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," *Proceedings of the British Machine Vision Conference BMVC*, 2014. 28

[74] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M. Yang, "Fast visual tracking via dense spatio-temporal context learning," *ECCV*, 2014. 28

[75] G. Roffo and S. Melzi, "Online feature selection for visual tracking," *BMCV*, 2016. 28

[76] A. Montero, J. Lang, and R. Laganiere, "Scalable kernel correlation filter with sparse feature integration," *: The IEEE International Conference on Computer Vision (ICCV) Workshops*, pp. 24–31, December 2015. 28

[77] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. Torr, "Staple: Complementary learners for real-time tracking," *CVPR*, 2016. 28

[78] M. Felsberg, "Enhanced distribution field tracking using channel representations," *Vis. Obj. Track. Challenge VOT2013, In conjunction with ICCV2013*, 2013. 28

[79] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," *ECCV*, 2016. 28, 52

[80] M. Tang and J. Feng, "Multi-kernel correlation filter for visual tracking," *ICCV*, 2015. 28

[81] C. Ma, J. Huang, X. Yang, and M. Yang, "Hierarchical convolutional features for visual tracking," *ICCV*, 2015. 28

[82] A. Lukezic, T. Vojír, L. Cehovin, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," *CoRR*, vol. abs/1611.08461, 2016. 28, 52

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012. 28

[84] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. 28

[85] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. 28

[86] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. 28

[87] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," *CoRR*, vol. abs/1704.03264, 2017. 28

[88] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 28, 42

[89] N. Wang, S. Li, A. Gupta, and D. Yeung, "Transferring rich feature hierarchies for robust visual tracking," *CoRR*, vol. abs/1501.04587, 2015. 28

[90] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 597–606, PMLR, 07–09 Jul 2015. 28

[91] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 28, 52

[92] H. Nam, M. Baek, and B. Han, "Modeling and propagating cnns in a tree structure for visual tracking," *CoRR*, vol. abs/1608.07242, 2016. 28

[93] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, (Washington, DC, USA), pp. 3119–3127, IEEE Computer Society, 2015. 28

[94] Z. Chi, H. Li, H. Lu, and M. Yang, "Dual deep network for visual tracking," *CoRR*, vol. abs/1612.06053, 2016. 28

[95] C. Sun, H. Lu, and M. Yang, "Learning spatial-aware regressions for visual tracking," *CoRR*, vol. abs/1706.07457, 2017. 28, 52

[96] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary learners for real-time tracking," *CoRR*, vol. abs/1512.01355, 2015. 29, 52

[97] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. 42

[98] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015. 42

[99] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. 43, 44, 46

[100] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1–1, 09 2015. 48

[101] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 48

[102] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: efficient convolution operators for tracking," *CoRR*, vol. abs/1611.09224, 2016. 52

[103] P. Senna, I. N. Drummond, and G. S. Bastos, "Real-time ensemble-based tracker with kalman filter," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 338–344, Oct 2017. 52

[104] E. Gundogdu and A. A. Alatan, "Good features to correlate for visual tracking," *CoRR*, vol. abs/1704.06326, 2017. 52

[105] T. McLaughlin and M. Hosenball, "Boston bomb suspect spotted on video, no arrest made," *Reuters*, Apr 2013. 54

Vita

Elliot Greenlee was born and raised in Knoxville, Tennessee. He attended Webb High School before starting college at the University of Tennessee in Fall 2013. During his bachelor's degree, Elliot gave tours as an engineering ambassador, studied abroad in London, and worked multiple internships, including a semester in Dallas, Texas. He also helped to start VolHacks, a student run hackathon on campus. For his senior design project, Elliot worked on a drone which could navigate indoors by following a person.

After earning a degree in computer science in 2016, Elliot began work on his master's degree as a Bodenheimer Fellow, focusing on machine learning and computer vision. During this time, he also worked at Oak Ridge National Labs as a researcher on a forensic security project. Elliot has presented work in New York City, Washington D.C., and Sydney, Australia. He is moving to Nashville to start work this summer.