Tracking and Measuring Objects in Obscure Image Scenarios Through the Lens of Shot Put in Track and Field

Ashley N. Smith

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Creed F. Jones, Chair
Hoda M. Eldardiry
Christopher L. Wyatt

April 25, 2022
Blacksburg, Virginia

Keywords: computer vision, object tracking, object measurement, deep learning, binocular stereo vision, sports

Tracking and Measuring Objects in Obscure Image Scenarios Through the Lens of Shot Put in Track and Field

Ashley N. Smith

(ABSTRACT)

Object tracking and object measurement are two well-established and prominent concepts within the field of computer vision. While the two techniques are fairly robust in images and videos where the object of interest(s) is clear, there is a significant decrease in performance when objects appear obscured due to a number of factors including motion blur, far distance from the camera, and blending with the background. Additionally, most established object detection models focus on detecting as many objects as possible, rather than striving for high accuracy on a few, predetermined objects. One application of computer vision tracking and measurement in imprecise and single-object scenarios is programmatically measuring the distance of a shot put throw in the sport of track and field. Shot put throws in competition are currently measured by human officials, which is both time-consuming and often erroneous. In this work, a computer vision system is developed that automatically tracks the path of a shot put throw through combining a custom-trained YOLO model and path predictor with kinematic formulas and then measures its distance traveled by triangulation using binocular stereo vision. The final distance measurements produce directionally accurate results with an average error of 82% after removing one outlier, an average detection time of 2.9 ms per frame and a total average run time of 4.5 minutes from the time the shot put leaves the thrower's hand. Shortcomings of tracking and measurement in imperfect or singular object settings are addressed and potential improvements are suggested, while also providing the opportunity to increase the accuracy and efficiency of the sporting event.

Tracking and Measuring Objects in Obscure Image Scenarios Through the Lens of Shot Put in Track and Field

Ashley N. Smith

(GENERAL AUDIENCE ABSTRACT)

Object tracking and object measurement through video capture are two powerful capabilities that emerged from recent developments in computing and hold compelling potential when combined. Object tracking is the fusion of detecting an object and constructing its path traveled between video frames, while object measurement is based on accurate translation from two-dimensional pixel coordinates to three-dimensional global coordinates. Achieving high accuracy in tracking and measurement is challenging when applied to ideal objects in an image (i.e. large, distinct from the background, not occluded). This level of difficulty significantly increases with small objects, moving objects that generate motion blur, or objects that blend in with the background. An ideal setting to study performance of these techniques during imperfect scenarios is sports, where tracking and measuring fast-moving objects is at the core of its objective. In track and field, the current method to enforce rules in throwing events, such as shot put is by utilizing human officials, but this task can lack safety, accuracy and efficiency. As a substitution for the method of human measurement in throwing events, a stationary two-camera software system is developed to programmatically detect a certain object, predict the object's path, convert the object's image location to three-dimensional coordinates and measure the object's distance traveled. Outside of the system's direct application, this work focuses on general improvement of tracking and measurement of imperfect scenarios and transitioning from the common method of tracking or measuring many general objects to tracking or measuring one specific object as precisely as possible.

Dedication

This thesis is dedicated to my grandma, my Yiayia, for whom I promised I would earn my master's degree.

Acknowledgments

First and foremost, my sincerest appreciation goes to Dr. Creed Jones who helped me every step of the way through this work and this degree. Additionally, thank you to Dr. Abbott for his help with stereo vision concepts as well as my other committee members, Dr. Eldardiry and Dr. Wyatt for their machine learning guidance. Thank you so much to Roderick DeHart for the countless hours of technical help and quick responses. Thank you to Nathan Hall and Lisa Becksford from Virginia Tech Libraries for their guidance in proper citations.

The deepest thank you to every math, physics and computer science teacher I have ever had - I applied hundreds of concepts from middle school all the way up to graduate school. Specifically, thank you to my first computer science teacher, Tony Smith, my first college professor, Kris Jordan, for sparking a passion for creating through code and my two software mentors who were the first to take chances on me - Dr. Gary Bishop and Scott Sundahl. Thank you so much to Paul Knackstedt for helping me stay the course every week and reminding me to stop and look at the view every once in a while.

Thank you to my incredible teammates, coaches and lifelong friends for helping me balance two dreams at once, listening as I worked through challenging problems, and assisting in data collection. Thank you to Antonio for believing in me more than I believed in myself during the highs and the lows. Finally, thank you to my mother, Clare [1], for teaching me the definition of hard work and persistence, father, Lindsey, for teaching me how to think like a true mathematician, and my sister, Kelly, for her unwavering support near and far. I could not have finished my thesis without the help of all mentioned.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Object tracking and object measurement, in a sequence of images, are two powerful capabilities that emerged from recent developments in computer vision. While each method has extensive and varied applications individually, they hold compelling potential when combined. Object tracking is the fusion of object detection and path prediction, while object measurement is based on accurate translation from two-dimensional pixel coordinates to three-dimensional global coordinates. Achieving high accuracy in tracking and measurement is challenging when applied to the most ideal objects in an image (i.e. large, unoccluded, distinct from the background). This level of difficulty significantly increases with small objects, fast-moving objects that generate motion blur, or objects that blend in with their background. An ideal setting to study the performance of these two approaches during imperfect scenarios is sports, where tracking and measuring fast-moving objects is at the core of its objective. A robust camera software system is developed to programmatically detect a certain object, predict the object's path, convert the object's image location to three-dimensional coordinates, and measure the object's distance traveled. Applying the two computer vision techniques of tracking and measurement to sports contributes to the burgeoning paradigm of task automation that is relevant to a variety of domains.

Sport is defined as any "game, competition, or activity needing physical effort and skill that is played or done according to rules, for enjoyment and/or as a job" [2]. Sport is ubiquitous across the world, promoting competitive fervor, uniting groups from varying backgrounds, and teaching significant life lessons everyday. Sports influence the economy even more than the culture, earning billions of dollars in revenue in the United States annually [3]. One of the world's most universal sports is track and field, where individuals compete against one another to run the fastest, jump the longest, or throw an object the furthest. [4]. Because of the wide range of event types, the sport collectively caters to almost every type of athlete.

In the United States, over 1.1 million high school students participate in track and field annually, including almost 500,000 females, which is the highest number of female participation in any high school sport [5]. There are more than 3,000 clubs at the local level associated with USA Track and Field and more than 8,000 competitions are sanctioned each year [6]. The throwing events in track and field are shot put, discus, javelin, weight throw and hammer throw. Each event involves a different standard object thrown, referred to as the "implement", and distances range anywhere from 3 meters to 98 meters, depending on the event and skill level. The shot put implement, specifically, is a uniform, spherical object that mimics a ball, has a tendency to travel along a projectile path, and travels a shorter distance compared to other throwing events [7, 71] *(See Figure 1.1)*.

The current method to enforce rules in throwing events and achieve a high standard of accuracy and fairness utilizes human officials, ranging from highly-certified to parent volunteers, depending on the competition level. Their responsibilities are often overwhelming, including explaining rules, checking sports equipment, determining if the thrower stepped out-of-bounds, and chiefly, measuring the distance traveled by the thrown implement after determining the location where it makes initial contact with the ground. [7, 73]. Furthermore, they must execute their job while dealing with criticism and social pressure from the

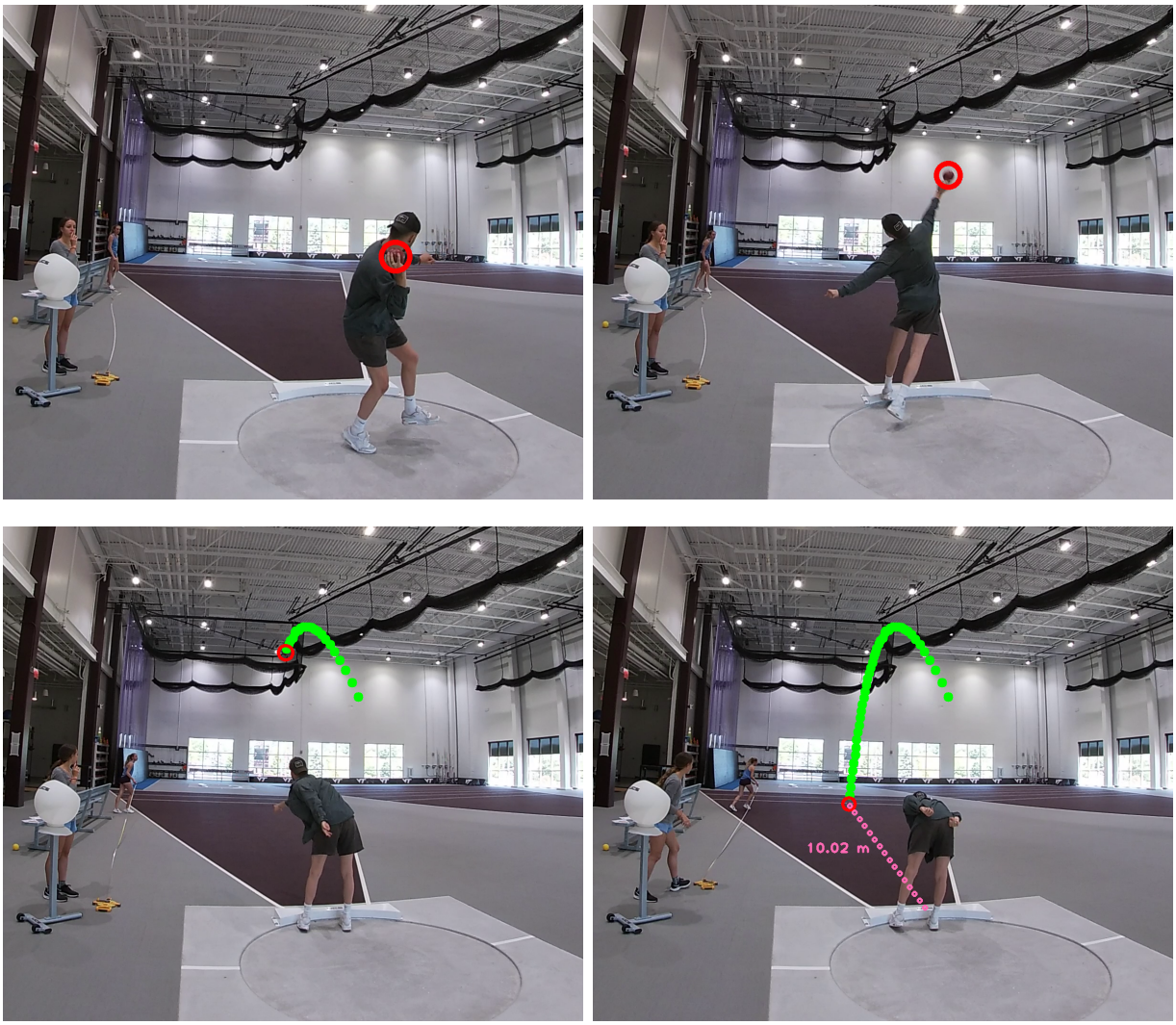


Figure 1.1: Four frames from the same video that collectively display a shot put throw in an official indoor throwing environment. In each frame, the red color encircles the shot put implement, and the green shows the path the implement has traveled. From left to right, top to bottom, the images show before flight, initial flight, mid-flight and landing of the shot put. After landing, officials move to the point where they believe the implement first made contact with the ground, and proceed to measure the distance of the throw from the edge of the stop board *(See also Figures 1.2 and 1.3)*.

crowd, raising the number of adverse conditions [8].

The first concern is this task puts officials at risk of being hit or impaled by these heavy objects, which can lead to serious injuries or even death [9] [10]. Second, the measuring

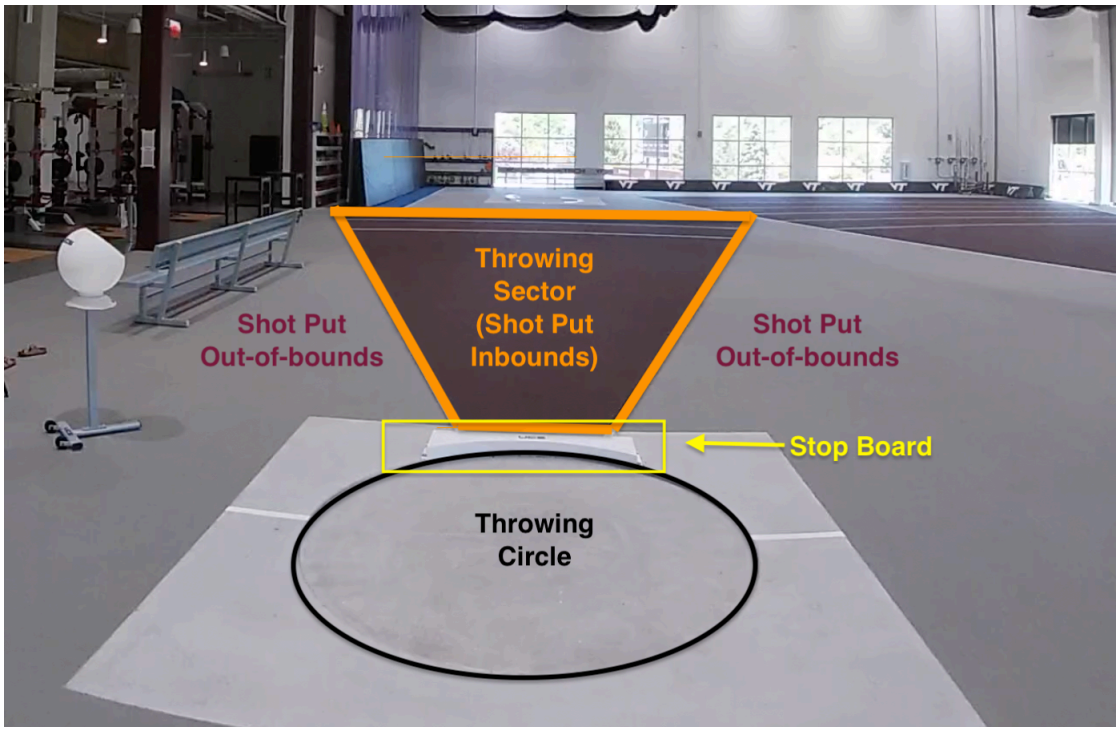


Figure 1.2: Labeled diagram of an indoor shot put throwing area. The thrower must release the shot put with their feet remaining inside the throwing circle. If the shot put lands inside the throwing sector, it is a valid throw, but if it lands outside, the throw is marked as a foul and the measurement does not count. The midpoint of the edge of the stop board that is closest to the throwing sector is the starting point for measurement.

process is both resource and time consuming, as it typically requires four judges and movement to the contact point (since they must stand far away during flight of the implement) followed by stretching a long measuring tape to record the distance [7]. Finally, this job can significantly lack accuracy, especially at the amateur level [11] for a plethora of reasons - the limited availability of certified officials, multitasking demanded by the job, physical limitations of humans, and even bias.

Multitasking for humans leads to decreased performance [13] and it becomes more difficult to maintain accuracy as the number of tasks multiply and the time frame shrinks. The human eye has a relatively limited frame rate [14], which restricts the processing speed of the eye and brain to accurately determine the landing point, especially when the implement

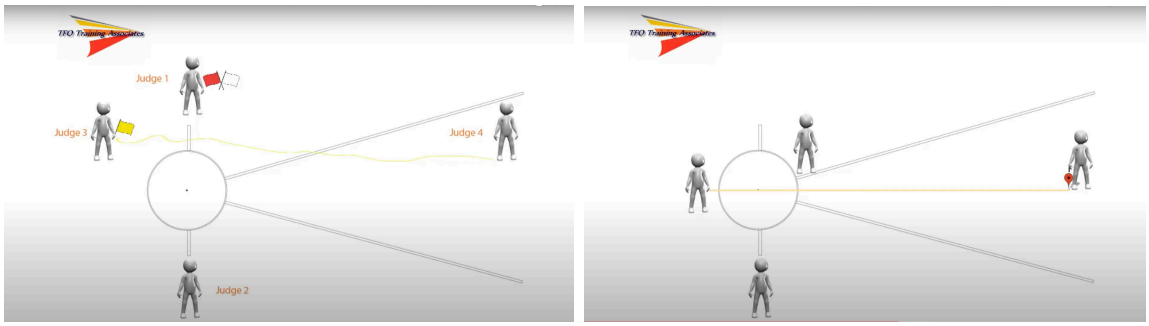


Figure 1.3: This diagram shows the position of each official (a.k.a. judge) during the shot put throw *(left image)* and after the throw has been completed to measure the distance traveled *(right image)*. Four officials are needed, which may be easily manageable at the elite level, but can be difficult to obtain at the youth and club levels. By implementing the system developed in my work in a competition environment, the number of officials can be reduced by at least half, and each remaining official will be in a safer position. *(Image credit: [12])*

moves at high-speed and officials are standing far away. Moreover, visual judgement can be hindered through obstructions, distance, height, shadow, parallax, depth and color [15]. Bias in referees can be prevalent due to factors, such as the desire to feel approval from the crowd, the stakes of the competition, the athlete's nationality or even the athlete's ethnicity [16]. In severe instances, referees may deliberately favor one outcome due to bribery payments or illegal gambling [17], which corrupts the integrity of the entire sporting competition. Ambiguities in measurement due to lack of knowledge and inconsistent techniques are also possible [11] - for example, the measuring tape might be positioned poorly or read incorrectly [12].

Erroneous measurements by officials lead to a lower level of standardization and are problematic because they can determine the result of the event, especially during elite competitions when the disparity between throws can be as small as 10 centimeters [18]. This can consequently lead to economic or psychological repercussions [17], affecting qualifications for championships, prize money, and even employment status of coaches and players [19]. To

mitigate the number of errors, keep officials safe and maintain the integrity of sport, an impartial, accurate and efficient software system should become ubiquitous at track and field throwing competitions.

Technological innovations have been present in contemporary sports for several decades – some pertinent examples are performance analysis through human activity recognition [20], movement screens to predict risk of injuries [21], and instant replay video for television broadcasts [22]. Beginning in 1999, instant replay video was adopted by the National Football League [19], where the coaches were given jurisdiction to request a video replay review to verify or retract the referees' calls. Instant replay has been commended by scholars [23] and has since been integrated into many major professional and some amateur [24] sports organizations. While beneficial, instant replay is costly [25] and time consuming [26], therefore it is not practical for track and field to employ instant replay technologies at most amateur levels of competition.

One technological advancement that was introduced into soccer in 2018 is Video Assistant Referee (VAR) [27], which is utilized to correct faulty calls made by officials in "match-changing" situations [28], like goals and penalties. An additional team of referees is located in an operations room, reviewing footage from the cameras after calls are made and communicating with the referee if necessary [29] . Like instant replay though, the ultimate call is still made by humans.

Systems exist in some sports, such as professional tennis, that use software to track objects in real time. For example, the 2021 Australian Open was played with "Hawk-Eye Live", a system using cameras to determine whether the ball lands in or out-of-bounds [30]. No human line judges were needed and several acclaimed players lauded the new technology, commending its high accuracy. The winner of the tournament felt the technology let the players focus on their performance rather than being constantly concerned about challenging

decisions from the official [30]. Object measurement techniques have been applied to other sports, like using the pinhole camera model and known object sizes to measure the displacement and velocity of a barbell in weightlifting [31]. Yet, there are few cases in the sports domain where both tracking and measurement are implemented.

In this work, an algorithm is developed to automatically measure the distance of a shot put throw in the sport of track and field. Because of the small size of the shot put, motion blur due to rapid movement, and its tendency to blend with the background, this work focuses on tracking and measuring less than ideal objects with respect to the camera. Insight into adjustments required to transition from the common method of tracking many general objects to tracking one to two specific objects is also provided, since the shot put is the only object of interest.

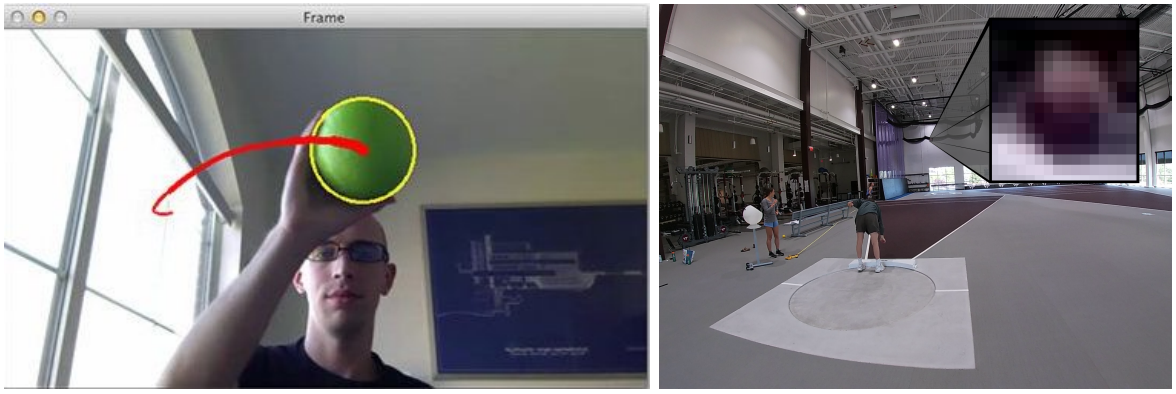


Figure 1.4: Visual comparison of an ideal scenario with a less than ideal scenario of the object of interest with respect to the camera. Both objects are round, uniform and ball-like. Yet, in the left image, the object is a different and distinct color against the background, it appears uniform, not warped by high speed motion, and its size is a high percentage of the entire image area - around 5.7%. *(Image credit: [32])*. In contrast, the right image shows the minuscule size of the shot put as it travels away from the camera - around 0.03% of the image area. When looking at the magnified pixel region around the shot put, one can see how it blends in with the background and appears warped.

My algorithm is comprised of deep learning and spatio-temporal pattern recognition to track the object and stereo vision principles to measure the object, along with novel techniques like

presumptive object detection to improve accuracy. This work is a step toward fully automatic tracking and measurement in sports, which reduces the number of required officials, thus increasing accuracy and impartiality of rulings, promoting safety, and lowering competition costs. It is also formulated on the basis of simplicity and cost-effectiveness, with the goal being an economical addition to amateur throwing competitions.

Results of this work can be applied to moving objects in other sports, such as measuring the distance traveled of a baseball or serve as a starting point for detecting rule violations in basketball or soccer. Outside of sports, tracking and measuring less than ideal objects is crucial to solving imperative problems in the world, such as identifying tumors in medical imaging [33] or pedestrians in self-driving cars [34], thus this work can also be generalized to benefit a range of industries.

1.2 Main Contributions

A stationary, binocular stereo system is proposed that performs object tracking through deep learning and path prediction followed by object measurement through three-dimensional reconstruction using two-dimensional image point correspondences. The main contributions of this work are:

1.) Create an end-to-end software system that has the potential to be a more precise, faster, and economical substitution for the method of human measurement for throwing events in the sport of track and field

2.) Demonstrate the value of improvements to detection and tracking, such as data augmentation and predictive path modeling, in scenarios where the object is minuscule, blends with the background, or is warped by high-speed motion

3.) Suggests adjustments to both tracking and measurement techniques when there are only one to a few object(s) of interest in an image as opposed to detecting and reconstructing many objects in the entire scene

4.) Error analysis of various factors that contribute to accurate and consistent triangulation from two-dimensional to three-dimensional points

1.3 Thesis Organization

Chapter 2 reviews established object tracking and object measurement concepts, as well as techniques to detect the specific objects relevant to my work. It also highlights examples of software applications in sports to explain why my selected methods are the best choice for this application.

Chapter 3 details the elaborate process of collecting data, calibrating cameras, and tracking and measuring the shot put in real time. This chapter includes descriptions, workflows and formulas of all approaches.

Chapter 4 assesses both quantitative and qualitative results and provides in-depth comparisons of various techniques applied to my work. It also includes shortcomings of current methods and suggestions to improve object tracking and measurement, both in general and in my specific application.

Chapter 5 summarizes the work and lists next steps toward research and development in the field of computer vision.

Chapter 2

Review of Literature

2.1 Computer-Assisted Technology in Sports

Presently, the two principal ideas for utilizing computer software in sports are processing and interpreting data from sensor-based systems or from camera footage. Both approaches apply some combination of machine learning and computer vision techniques on the vast data garnered from sensors or image sequences. As these fields continue to advance, the potential for accurate and reliable technologies in sport grows, and the ability to seamlessly track objects of interest becomes an attainable objective. Sensors collect a plethora of biometric data including body movements [35] and videos contain a myriad of pixel-wise information. While both concepts have advantages and disadvantages, for reasons that will be addressed, the method that interprets data from video footage is the more prudent solution.

Within sports, the predominant subset of data processing from sensors is wearable technology, defined by any "small computer or advanced electronic device that is worn or carried on the body" [36]. These devices, also called wearables, have recently permeated the sports world because of the data-driven ability to analyze performance, reduce risk of injury, and monitor one's physical health. One relevant application of these devices and their associated data is to determine any illegal moves by monitoring movements of athletes during competition. For example, Taborri et al. conducted an analysis on detecting faults in race walking through wearable devices. Seven inertial sensors were placed on the lower limbs of each test

subject and angular velocities and linear accelerations were calculated from this data to characterize rule violations from the lower legs [37]. Likewise, Chi of Palo Alto Research Center employed "piezoelectric force sensors" to assist judges in scoring tournaments in taekwondo [38]. These products did demonstrate sufficient accuracy, but for the following reasons, mass acceptance of wearables for assisting officials is not practical in the sports industry.

First, wearable devices often experience malfunctions, including frequent drift errors from inertial sensors [39]. Wearables also have the potential to be highly uncomfortable and could injure the athlete or his opponent [40]. Athletes move in more unique and frequent manners than average users, and wearables may cause athletes to change their motions and gestures in a detrimental manner. An example of this issue is the kneepads with strain transducers invented by Ciu et al. for illegal steps in race walking which unintentionally altered their walking form [41]. Furthermore, many athletes are concerned about how the vast amount of personal data collected from wearables is being used by coaches, managers and wearable companies [39]. In some extreme instances, wearables may even force changes in the rules of the sport [38]. Because of these numerous factors, many athletic professionals do not trust their efficacy [39], therefore it would be difficult to build a system from wearables that will be eagerly adopted by various governing bodies of sports.

A smaller subset than wearable devices, sensor technology also includes sensors installed in the environment for which the sport is played. For example, in bowling, a sensor is placed underneath the lane to detect if the player steps over the foul line, as this is a common violation in the sport. This procedure has been perfected to accurately make calls and send alerts without interrupting the players [38], which is a paramount element of any successful computer system in sports. While this functions well in small environments like bowling, it is not the most feasible solution for larger play areas, and could be a large expense to install. The system does not thrive in sports with constant motion because it distracts opposing

players, and for these reasons, this type of technology has also not been readily adopted by sports associations [38]. In the throwing events, sensors could be installed in the implement to measure its distance traveled; however, this runs into the same issues of potential cost, which may be feasible at the elite level, but unlikely at the youth level. Additionally, it would be arduous to standardize due to the fact every individual or team uses their own implements to train and compete.

Conversely, camera technology and its coupled software is an alternative and fruitful method for detecting rule violations or assisting judges in sports. There are a substantial number of cameras set up for most professional sports and at a range of angles, thus eliminating the need for additional equipment and reducing the number of changes to the ambience of the sport. For example, the National Basketball Association (NBA) invested in SportVu [42], which placed six very high resolution cameras in every professional arena that each collect data 25 times per second. HawkEye systems, prevalent in tennis, also utilize six cameras to detect and track the ball from multiple angles [15]. Underwater camera systems have been installed and recently approved by the governing body for six aquatic sports to be utilized like instant replay. Judges can access underwater video footage to review the stroke and revoke a disqualifying call [43]. Instant replay and VAR are two examples of applications which already employ many cameras [22] [27].

Video footage has been deemed the best solution due to recent breakthroughs in computer vision and machine learning techniques, enabling the development of algorithms to make in-depth, real time conclusions from a sequence of images. It is also a more feasible path because cameras can be positioned at reasonable distances from the playing area so as not to interfere with the athletes or supporting personnel. Specifically, in throwing events of track and field, the cameras can be placed far enough away to avoid being hit and damaged by heavy implements. Additionally, filming of sports competitions began in 1939 [44] and

has become ubiquitous among professional sports today. While the algorithms may be new, the videotaping of athletes is not, so changes to the surroundings will be minimal. The following sections will review various concepts and technologies related to object tracking and measurement that precede and contribute to the development of my computer vision system for the shot put event.

2.2 Object Detection and Tracking

There are three components amalgamated in the detection and tracking section of my work - the shot put, the thrower, and the environment. Accurate and efficient detection of each entity requires a combination of specific machine learning and computer vision developments. Spatio-temporal pattern recognition of all three together leads to the ability to track where the shot put flight is initiated and completed, and determine if the throw is legal. Currently, many computer vision detection algorithms focus on pattern recognition for singular objects, but the more complex problems require multi-object pattern synthesis.

2.2.1 Machine Learning Techniques

The rapid improvement of machine learning techniques based on artificial intelligence and statistical methods has led to the advancement and higher accuracy of classification problems. Classification is a vital foundation for detecting the shot put, the thrower and any other pertinent aspects of the environment, such as the bounds of the throwing sector. Supervised learning is a subset of machine learning used in classification where the instances are provided with known labels, and features for training are selected from the data to distinguish classes. It is the most popular machine learning technique [45] as it is utilized across a large

range of industries due to its simplicity and specificity of boundaries. Several studies involving machine learning in sports demonstrate a preference for supervised learning algorithms [46]. In basketball, the technique was employed to train the Second Spectrum software to detect shots, passes and rebounds as well as the pick-and-roll play [47]. Second Spectrum also helped the Los Angeles Clippers, an NBA team, release CourtVision, which augments visualizations and statistics on top of video footage through machine learning processes [48].

One of the most common machine learning algorithms is the nearest neighbor algorithm, specifically k-nearest neighbor (kNN), which was created with the idea that data points with the most similarities in a feature space lie close together, therefore belonging to the same classification. kNN is simple, yet powerful and incredibly stable. It also has disadvantages, such as high storage requirements of all data points and generally more computation time during classification [49]. This idea is also prevalent in various computer vision algorithms, such as the use of kNN for background/foreground segmentation [50].

A more recent technique, artificial neural networks (ANN) were developed as a way to classify instances in situations not linearly separable. They are a "multi-layer network" that consists of three types of connected units - input, output, and hidden. Challenges to a successful ANN are finding the ideal size of the hidden layer and properly training the weights of the network, the most defining part of the algorithm. Defining the weights is typically done through the Back Propagation (BP) algorithm, which invokes the chain rule and iterates backwards to compute the gradient by layer. ANNs are complex and often work well for intricate data sets from real world scenarios, yet have a questionable robustness on their own due to high variance and unsteadiness [49].

Challenges in machine learning algorithms overall are poor approximation, known as underfitting, generalization, or overfitting [49]. Increasing the number of features or dimensionality of the data does not necessarily lead to a better algorithm in several sports examples [37]

[51], thus a crucial step is to find a middle ground number of features. For small objects like the shot put; however, more complexity may lead to improved detection and tracking.

Additionally, there are supplementary methods to account for incomplete data, which is likely required in a dynamic and chaotic sporting environment. Techniques for inferring missing feature values of data points include selecting the most common feature value in the class, inserting a feature's mean value or regression and classification methods [49].

Open source tools within machine learning are heavily used in large scale applications, and include TensorFlow [52] and PyTorch [53]. TensorFlow specializes in training deep neural networks and PyTorch is typically applied towards computer vision problems involving machine learning.

To validate the efficacy of a machine learning algorithm in a specific scenario, there are several techniques that have been developed, namely cross-validation for supervised learning. Cross-validation is a data resampling method that aims to catch any overfitting errors or erroneous generalizations. Repeated sampling occurs where a subset of the data is utilized for training and the remaining subset is then applied for testing. It is an assistive tool for not only selecting the appropriate algorithm, but also optimizing the parameters, like the ideal k value in kNN [54]. Precision, recall and F1-score are also useful for testing. Validation techniques will be applied to verify the legitimacy of the object detection results in my work.

2.2.2 Video Analysis Through Computer Vision

Computer vision techniques originated from manipulation and analysis of single images, but have now extended to video, which at its core is just a sequence of images. Video processing in tandem with machine learning allows for deep analysis of multiple images where their spatio-temporal relationships determine the conclusion of the situation. Results from video

analysis tell users both the content of each image in the sequence and the action occurring throughout. Image processing, at the simplest level, has three main steps - "classification, localization and object detection." The challenges of image processing are accuracy, speed and cost [55], and these issues are only magnified on the video level as faults in each image compound over an entire video. Most of the time the data from the individual images is insignificant without tracking trajectories to a high precision [56].

The Convolutional Neural Network (CNN), presented by AlexNet in 2012, was a major development in the nexus of computer vision and machine learning, boosting the valuation of video processing [57]. CNN reduced the error rate to 37.5% and 17.0% for top-1 and top-5 error rates, respectively, in the ImageNet LSVRC-2010 contest. This major improvement consequently sparked more development in deep learning. Following this, Regions with CNN (R-CNN) developed new ways to identify different segments on an image through support vector machine (SVM) classification and bounding-box regression. Fast R-CNN and Faster R-CNN decrease computation costs and greatly reduced overall computation time. ResNet (Residual Neural Network) was created with an error rate of only 3.6%, which is the first to be less than that of the human eyes (5.1%) [55].

Building on existing CNNs, a new algorithm was developed called You Only Look Once (YOLO). The idea behind YOLO is to perform only one analysis of the image by detecting objects through regression on independent bounding boxes and their class probabilities (predicted with a single neural network). Because of its efficient design and simultaneous computations, it is a real time detection system and additionally maintains respectable accuracy [55]. This system employs Darknet, a C implementation of CNN [58] and utilizes the K-means clustering algorithm to classify dimension groups of the bounding boxes [59]. It is pre-trained with the acclaimed COCO (Common Objects in Context) dataset from Microsoft, which contains over 200,000 labeled images and has 80 object categories [60]. Five

current versions are publicly available, providing both pre-trained weights and the code to train custom datasets. There also exists a Fast YOLO which focuses on fast object detection and works at 155 frames per second (FPS), even better for minimizing computation time [55].

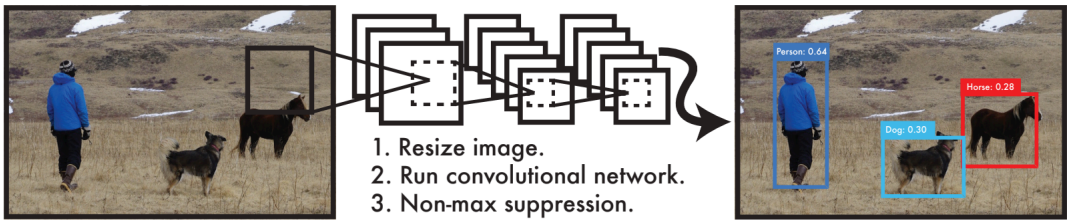


Figure 2.1: A "simple and straightforward" diagram of the steps that YOLO takes to detect objects in an image in real time. The image is resized to an image of 448x448 dimensions, only runs one CNN, then one final postprocessing step. *(Image credit: [61])*

YOLO has several benefits as it maintains a balance between speed and accuracy, and works as a real time detection system, outperforming both Faster R-CNN and ResNet. Yet it does struggle to recognize small objects in a group and identify the same objects when only their positioning has changed. YOLO has room for improvement regarding occlusion as it lacks the ability to make inferences when objects are not clearly exposed in the image frame [59].

YOLO has been implemented for research in several domains, such as analyzing traffic in the transportation industry [62], distinguishing identical animals in the farming industry [63] and detecting players [59] and accompanying objects [64] in the sports industry. One recent study from 2019 utilized YOLO to track the passing order of the ball in basketball and consequently synthesize pass networks from the results. This algorithm both tracks the ball and identifies players through Darknet. Their first YOLO model only has two classes for its bounding boxes to distinguish between a player who is holding the ball versus not. Additionally, these researchers modified YOLO by processing information from adjacent

frames to improve its detection accuracy when several objects are occluded. In a sports setting, players and objects are in constant motion. The results of this study are beneficial for my work because they focus on making improvements in highly dynamic and chaotic scenarios where players leave frames or obstruct the view of each other - there is no guarantee the shot put will be unoccluded and identifiable in every frame. It is an important step in the direction of intelligently making inferences and assumptions when objects are not perfectly present, which is a likely scenario both in and out of sports.

2.2.3 Shot Put Detection

Tracking the shot put is paradoxically a straightforward and complex problem. Many sports involve only one ball that is the focal point of the event, making it easy to track since there are no similar-looking objects. However, in those same sports, the ball travels at fast speeds in infinite directions, making trajectory a challenging problem. In tennis, hitting speed averages around 75 mph for both female and male players [65], a quarterback can throw a football up to 60 mph [66], and the average Major League Baseball (MLB) player throws a fastball at 92.3 mph [67]. Additionally, tracking systems must be robust enough to handle occlusion from players and spectators, as well as differentiating the ball from background objects, such as birds flying overhead [56] or flickering lights in indoor settings.

HawkEye has emerged as the primary provider of automatic ball tracking in sports. Launched in 2001 by computer expert, Paul Hawkins, it is composed of high-speed cameras capturing more than 200 FPS, computers equipped with real time imaging technology and electronic screens [15]. It was introduced in an auxiliary role as assistant referee technology, but over time, has acquired more responsibilities to the point where it can now operate independently of any line judges in tennis [30].

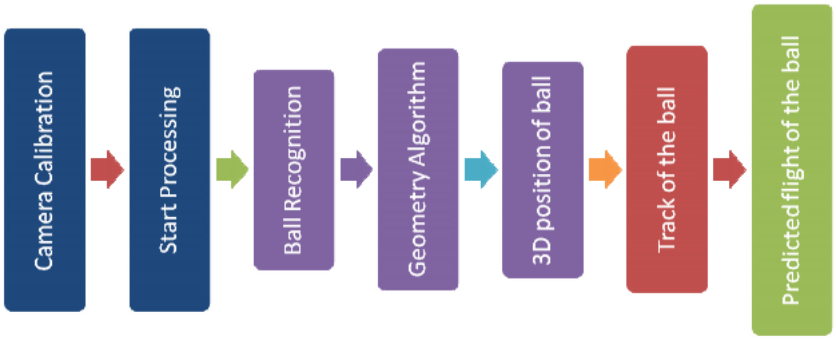


Figure 2.2: The image above outlines the process developed by experts at HawkEye. Initially, the system begins with camera calibration. After the ball is found in the image, Hawkeye employs a geometric algorithm to calculate the three dimensional coordinates of the ball from analyzing multiple images. Once multiple coordinates are obtained, a curve of motion is drawn to represent the flight of the ball. The image processing portion has three main steps - determining the pixels of the ball, applying the geometric algorithm to a set of images, and determining its three dimensional position. *(Image credit: [56])*

Some beneficial aspects of HawkEye include its switch between active and passive mode in order to save computations, its ability to track statistics [56] and engage spectators [15] and its high accuracy that referees cannot reach. In cricket, its detection of a 'leg before wicket' violation reaches 99.9% accuracy. Its predictive modeling of the trajectory of the ball can predict points of contention, like if the ball lands out-of-bounds in tennis or hits the pitch or stumps in cricket [56]. This predictive power can serve as a backup system in the case that for various obstruction or speed reasons, the actual position of the ball is not recorded. While HawkEye has experienced success in tennis and cricket, the technology has not yet been commercialized in track and field, nor to measure the distance traveled of the shot put.

Predictive modeling is a powerful form of backup for tracking that applies spatio-temporal context based on common kinematics formulas, including measuring the speed of movement of a barbell in weightlifting [68] or identifying the correct path of a volleyball [69].

2.2.4 Small Object Detection

As the shot put moves further away from the stationary camera, its size with respect to the image significantly decreases, eventually becoming a small object detection and tracking problem. Even though object detection has experienced impressive results, it is heavily skewed towards the identification of medium- and large-sized objects, while small object detection severely lags behind [70]. Small objects present a deeper challenge because they are composed of only a modicum of pixels - MS COCO defines small objects as an area of 32x32 pixels [60], but many objects are much smaller - tiny objects are defined as 16x16 pixels or less [71]. Usually less than 0.5 meters per pixel in resolution, small and tiny objects are often mistaken for frivolous noise [72]. This becomes even more common in techniques utilizing CNNs due to its high number of downsampling operations in the middle layers, thus never being present in the detection stages [73]. Increasing the resolution of images is a prudent idea, but it causes an exponential increase in computation time and memory usage that is often not worthwhile [73]. Even with these changes, many times, small objects still go undetected in high-resolution [70]. Additionally, many datasets do not contain enough instances of small objects, therefore training is hindered. For example, in the established MS COCO dataset [60], only half of the training images contain small objects and furthermore, only 1.23% of the total number of labeled pixels corresponds to small objects [70]. Together, all of these conditions make it difficult to garner enough semantic information during training and testing to adequately identify the small objects of interest [72].

Mitigating this problem is valuable because there are several pertinent applications for small object detection, in some cases, more important than identifying the largest objects. Along with tracking a sports ball, use cases include objects in infrared images for the military [74], early detection of tumors in medical imaging [33], and recognizing small objects to avoid on the road in self-driving cars [34].

Many existing works on small object detection focus on methods that improve overall object detection, regardless of the object size [70] [73]. While this is a valid problem to solve, there are many use cases where the only focus is detecting small objects, indifferent to any effects of detection results on larger objects. Hence, there is also a need for methods concentrating on isolated small object detection.

Data Augmentation

Data augmentation, a method "to significantly increase the diversity of data available for training models, without actually collecting new data" [75] is one method to alleviate the paucity of small object training data. Yet, it must be performed carefully to avoid constructing images that are too different from reality. Methods for augmenting data can be divided into two categories - those that involve pixel adjustments on the entire image, such as increasing brightness or adding Gaussian blur [76], and those that are more targeted and application-specific. One example of a targeted strategy is copying and pasting the object of interest onto several locations in the image, which was tested in the paper "Augmentation for small object detection" by Kisantal et. al [70]. While the copy-pasting strategy improved the mean average precision (mAP) results for small objects, it did struggle with overfitting due to a lack of blending between the pasted object and its new background, causing the model to learn these examples easily. Simple data augmentations have been effective for improving image classification, regardless of the level of quality of the original data [77].

2.2.5 Thrower Detection

For many applications of computer vision in sports, it is vital that these algorithms understand more than just the basic tenets of human motions [78] to accurately track detected

movements, predict future movements, and infer movements that are occluded. Designed to exclusively identify humans, OpenPose is an open source system that focuses on detecting the two-dimensional pose of multiple people in real time [79]. It was constructed with a bottom-up approach utilizing Part Affinity Fields (PAFs), which is a set of two dimensional vector fields that record the orientation of limbs. Results prove it still maintains a high accuracy that is not dependent on the number of people in the image. OpenPose can also make inferences to detect occluded persons or distinguish adjacent people who have little physical space between them.

Many alternative algorithms for human detection in sports use bounding boxes [80] [59], but sometimes lead to erroneous detection of multiple players within the same box. YOLO is also capable of detecting humans through the bounding box method, as a 'person' is one of the 80 classes of the pre-trained model. This may prove accurate enough for the purposes of my application since only the thrower must be identified, and will likely be away from other humans.

Developers of this algorithm also created a combined body and foot keypoint detector, and released a now publicly available annotated foot dataset with 15,000 instances along with several hand and facial keypoints. These features are highly useful in training and testing because identifying the beginning of a throw can be dependent on the position of one's hand, closely followed by the position of one's feet to monitor if the thrower has remained inbounds. This is valuable because if the thrower is deemed out-of-bounds, there is no need to execute the subsequent measurement code.

Athletes are not the only humans present in this environment, and it may be important to differentiate them from the cadre of personnel like coaches, trainers and judges. To account for this, Santosh and Kaarthick implemented a color-based detection system for players which served to both rule out trivial "pedestrians" and also identify the team for which

Figure 2.3: The leftmost image above shows the results from detecting athletes during competition through the OpenPose system. In these examples, their stances are straight up and they are the only humans in the respective images. From (b), it can be seen that the ball is not detected and (c) does not detect alternative objects like the tennis racket. The next two images are common failures when body parts are occluded in the frame *(Image credit: [79])*

each player belongs. The color-based detector detects the player by creating a Histogram of Oriented Gradients (HOG) box, which is a subset of the image classified by the HOG detector. Performing this on the HOG boxes helps eliminate any color detection of the environment, such as court lines and spectator clothing. An HSV scale (hue, saturation, value) was selected to best discriminate between steep changes in color. Histograms were built from the HSV coordinates, thresholds were set and binarization was performed to distinguish between jersey colors, thus determining the team association [80]. Once the thresholds are set, the classification can be performed quickly; however, setting the thresholds may eliminate the real time potential of this method.

Initially, all humans were detected through the HOG detector from OpenCV, which implements HOG features as well as SVM classifier. While the HOG detector appeared to be the best choice, it still failed often and a redundancy system needed to be implemented, hence another reason for the color detector. In dynamic sports, there are several positions an athlete can take as they wind up their throw that are not easy to disambiguate, leading to an unfavorable error rate. Color detection could instead be a viable solution to alleviate

Figure 2.4: Above is a visualization of the process to distinguish players based on their jersey colors. This most crucial element of this process is determining the threshold based on the uniform colors of the opposing teams, which can be adapted to calibrate before the game, allowing the system to still operate in real time. *(Image credit: [80])*

occlusion problems as it can be used to identify the player in boxes from preceding frames [80]. While color-based detections worked in this scenario, it may not be useful for the shot put if the background colors closely match the object of interest.

Instead of color-based approaches, background subtraction approaches may work better, especially in situations where the camera is stationary and little is in motion other than the object of interest. Background subtraction capitalizes on the assumption that there is an obvious difference between the "current frame" and some "reference frame" that is periodically updated for natural shifts in light and camera position [81]. The thrower and shot put are most likely moving at a much faster rate than anything else in the frame, making background subtraction a potential element of my algorithm.

2.2.6 Environment Detection

Detecting the sports environment is the least imperative of the three, yet still a salient factor in this work. The environment encompasses anything from the throwing circle, its

boundaries and sections to the locations of spectators, coaches and medical staff. An early step in the research by Santosh and Kaarthick was to correctly identify the lines of the court *(See Figure 2.5)*. Some aspects of this process may be beneficial towards my work to identify out-of-bounds; however, some may require too much computational time to function in real time. Detecting various aspects of the environment may help determine certain rule violations, such as if the shot put is thrown out-of-bounds or the player illegally steps outside of the throwing circle.

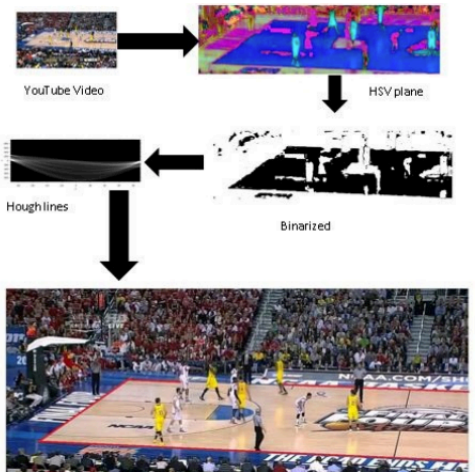


Figure 2.5: The process of utilizing open source computer vision techniques to detect the court boundaries. They first converted the images from the RGB (red, green, blue) scale to the HSV scale and performed erosion and dilation on the image to eliminate trivial objects unrelated to the court. They then employed the Canny edge detector and Hough Transform, both publicly available operators, to determine the straight lines in the system because they represent the boundaries of the court. *(Image credit: [80])*

2.3 Object Measurement

Measuring the distance of a shot put throw requires accurate reconstruction of three-dimensional world coordinates from their corresponding two-dimensional pixel coordinates. The only two points that matter are the shot put landing and the edge of the stop board, as these are the two ends of the measurement. Therefore, this system does not depend on the knowledge of depth and size of all objects in the image, which is the case for many three-dimensional reconstruction applications, such as those for self-driving cars [82] and land surveying [83].

Due to projective ambiguities, converting from two dimensions to three dimensions requires knowing the location of the same point across multiple images. Any number of cameras greater than one can be used to identify the three-dimensional position of an object; however, a binocular stereoscopic (two camera) system is one of the most common, as this "resembles the basic mechanism of the human eye" and is cheaper and more reliable than many three-dimensional sensing systems [84]. From the stereo images, there are three common methods to perform reconstruction, which are all possible since two images "contain enough information about the camera position and about the position of scene points relative to the camera" [85] for "trigonometric principles" to be applied [86].

The most promiment method is calculating the mathematical relationships between two images, which follow the pinhole camera model, using steps from Hartley and Zisserman [87] based on epipolar geometry. The first step is to estimate the fundamental matrix through the 8-point algorithm [88], which requires eight or more point correspondences between images. Point correspondences can be obtained manually, or through various established detectors, including Harris corner detector [89], Scale Invariant Feature Transform (SIFT) [90], and Oriented FAST and Rotated BRIEF (ORB) [91]. The essential matrix is then extracted by multiplying the fundamental matrix by the intrinsic matrices of the two cameras. Once

the essential matrix is found, it is decomposed into two possible rotation and two possible translation matrices. Only one combination of rotation and translation matrices results in the correct projection matrix, which is then used to triangulate the two corresponding pixel coordinates into one three-dimensional world coordinate from the reference frame of the left-most camera. *(See Figure 2.6 to visualize steps).* There are several functions in OpenCV [92] that perform different computations of this process, depending on the parameters known ahead of time.

The algorithm relies on a strong camera calibration and point correspondences, as most results stem from the fundamental and intrinsic matrices; however, a major benefit is it does not require knowledge of the baseline of the two cameras, nor require the cameras be positioned in any specific configuration. It also does not require the depth of all points in the image to be known.

Another method for reconstruction is creating a depth map for the points among the two images, which then returns disparity values that are used in "distance estimation of the object", given that depth is the inverse of disparity. Using calibrated cameras, Semi-Global Block Matching algorithm is a common way to find point correspondences and construct a depth map [86].

The benefit of this algorithm is that the depths are well-known; however, in applications where the only focus is on a few specific image points, this may require more time than is worthwhile. Additionally, it requires several passes of fine tuning and trial and error to acquire an acceptable depth map.

One final reconstruction method involves "rectification", which is the "act of projecting two stereo images onto a common plane" [96]. Rectification simplifies the equations to transition from a two-dimensional to three-dimensional point, given that the epipolar lines become

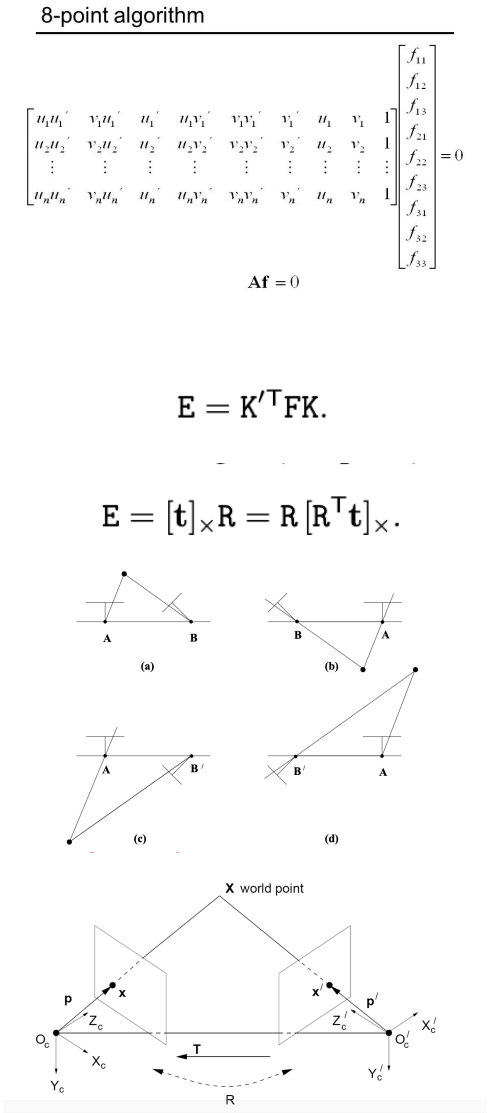


Figure 2.6: Mathematical process of point correspondences between the two images to extract the world point. The first image shows how known point correspondences make up the fundamental matrix, the second image relates the fundamental and essential matrices from the intrinsic matrices of the left and right cameras (K). The third image shows the extraction from the essential matrix to rotation and translation, while the next shows how only one of the four possibilities is correct (a). The final image shows the idea of triangulation to obtain the global point once all relationships between the two image points are known. *Image Credit: [87] [93] [94]*

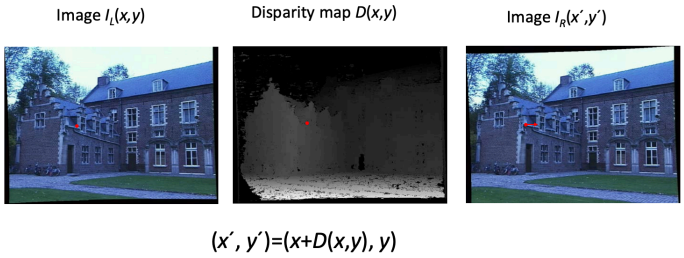


Figure 2.7: Visual example of a depth map created from two stereo images. The depth map then leads to the ability to calculate disparities, which is useful to solve the global points in Equation 2.8. *Image Credit: [95]*

"collinear and parallel to one of the image axes" [97] which in most cases is the horizontal, y-axis. Based on a method for stereo pairs proposed by Fusiello, Trucco and Verri [97], a linear rectification algorithm exists that generates projection matrices for each camera. It is typically performed to make the discovery of point correspondences between the images easier to find, as it reduces the search to only one dimension and can be used to solve reconstruction problems as well. Rectifying the images, followed by Equation 2.8 can result in an accurate measurement, but does require a known baseline.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{B}{d} \begin{bmatrix} x'_L \\ y'_L \\ f \end{bmatrix}$$

Figure 2.8: The equation used in parallel stereo systems to calculate a global point from world points, which also works on rectified images. B is the baseline, d is the horizontal disparity, f is the focal length, and x'L and y'L are the pixel coordinates of the left images. x, y, and z are the components of the global coordinate. *Image Credit: [94]*

All methods rely on some number of known point correspondences between the two images, which may not always be known ahead of time. Thus, the correspondence problem, is one

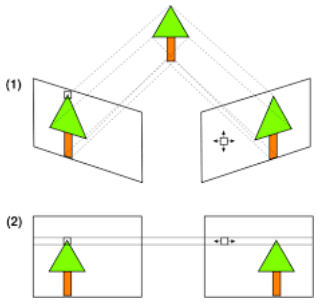


Figure 2.9: Two rudimentary images that point to the same global point before and after the rectification process. One can see in the rectified images how the y-axis lines up in each image, making it easier to find point correspondences. *Image Credit: [98]*

reason reconstruction can be challenging in real-world applications. To verify the fundamental matrix (F), one can measure the error based on the fact that a point in the right camera multiplied by F multiplied by the corresponding point in the left camera equals 0 ($x_{left} * F * x_{right} = 0$). The same process can be done to verify the essential matrix, but with normalized coordinates instead.

All methods also rely on an accurate camera calibration to determine the intrinsic parameters, including focal length, location of the image center and any lens distortion parameters. Zhang's checkerboard calibration is a common method to find parameters necessary to undistort the images [99]. Reconstruction is only accurate up to a certain scale ambiguity from the images alone [85], therefore, in order to scale the world coordinates to metric units for measurement, one must know the focal length, as the scale factor equals the inverse of focal length [100].

Stereo vision can be computationally expensive due to its high number of repetitive operations; however, when prioritizing accuracy over efficiency, it is a prominent solution. It is a common technique applied to many computer vision applications for the purposes of distance measurement, including avoiding obstacles in self-driving cars [101] and an alternative technique for surface imaging in medicine [102]. However, many existing applications are

concerned with the distance between the camera and the object, rather than the distance between two objects of interest both present in the images. In sports, specifically, the idea of stereo vision has been used to determine the speed of a ball for alternative racket sports [103] and measuring an athlete's displacement [104]. Each method for reconstruction based on a pair of stereo images has advantages and disadvantages, and will all be evaluated to determine which yields the best results for the purpose of this specific application.

2.4 Review

This work aims to bring all of the previously mentioned techniques together and build from existing systems to create an algorithm whose focus is tracking through multifaceted spatio-temporal pattern recognition and measurement through stereo vision, with a secondary focus on accuracy in real time and occluding scenarios. Multiple common object detection and tracking techniques will be tested. Moreover, several computer vision techniques and open source libraries will be employed like YOLO, OpenPose and OpenCV. Finally, the best way to detect and track will be chosen, and the best way to reconstruct points for measuring distance will be chosen, and the two will be compounded for optimal tracking and measurement of a shot put throw.

Chapter 3

Methodology

Automatically measuring the distance of a shot put throw is divided into two principal components - the deep learning plus kinematics component to track and determine the shot's landing point, and the stereo vision component to measure the distance from the stop board to the landing point. The object tracking component was initially built using background subtraction and filtering connected components; however, due to the small size of the shot put as it moved further away from the camera, as well as its common case of blending in with the background, deep learning techniques were applied by training a convolutional neural network. Three techniques were tested for the measurement component, but the Hartley-Zisserman fundamental matrix technique proved to be the most reliable, with the addition of obtaining manual point correspondences and refining them to a sub-pixel accuracy. Before running the real time tracking and measurement algorithm, first, data had to be collected, and the camera and scene were both calibrated.

3.1 Data Collection

Because there are no publicly available, comprehensive datasets of shot put throws, and several known measurements of camera position are necessary for the work, I procured my own data. Five individuals were recorded at various distances and angles throwing a standard-size competition shot put in the throwing area inside Rector Field House at

Virginia Polytechnic Institute and State University. Two GoPro Hero 7 Silver cameras [105] were utilized, either mounted on the same tripod a small distance apart or mounted on separate tripods a larger distance apart. Regardless, the cameras were always positioned as parallel to each other as possible. Both cameras recorded all videos at a dimension of 1920 x 1440 pixels using lenses with a focal length of 33 mm. The height of the cameras was tall enough so the entire trajectory of the throw was in the field of view - the cameras never moved to follow the shot put, but instead remained stationary, which is a crucial design decision of my system.

In total, 54 videos were recorded (27 from each camera) at a frame rate of 30 frames per second, and each video was around three seconds in duration. 24 videos contained full throws of the shot put, while three videos captured potential edge cases of the shot put or thrower. Several different camera positions were tested - centered, left and right of the throwing circle, both behind and parallel to the thrower. Multiple shot put implements or placeholder balls were used in testing, each with different colors and sizes, as well as scenarios with the lights on and dimmed. Appendix A provides a full table of each video and its associated metrics.

From these videos, several less than ideal scenarios were produced, such as frames containing severe motion blur of the shot put or backgrounds that blended with the color of the implement. For example, frames showed the shot put at an extreme contrast against natural light, dark backgrounds against textured nets in the rafters and the floor, and neutral backgrounds in front of the gray wall. There were also numerous frames that represent partially or fully occluded scenarios with respect to the shot put - hands, hair, or bodies blocking its full representation *(See Figure 3.1)*.

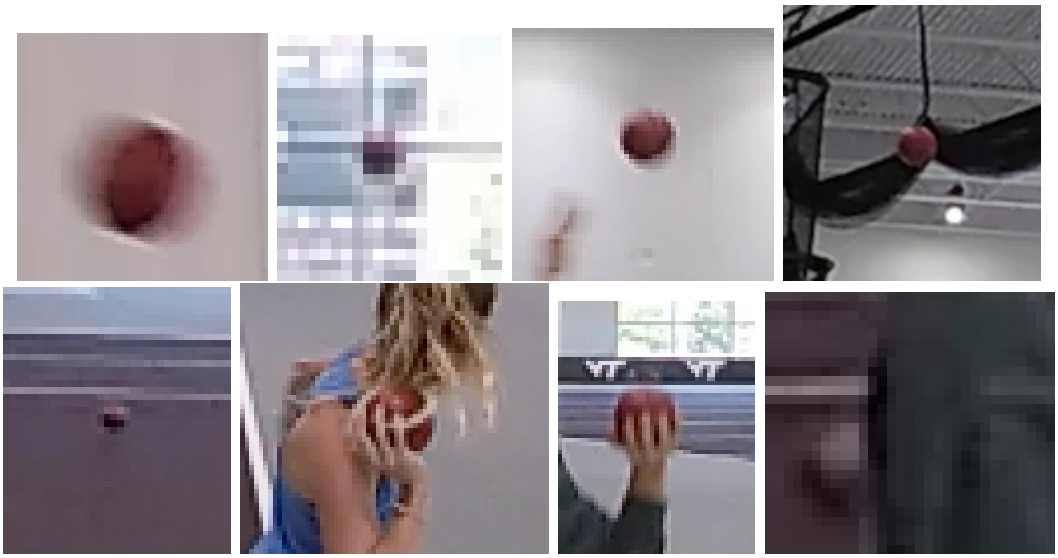


Figure 3.1: Examples of diverse shot put appearances from frames obtained in data collection. From left to right, top to bottom - motion blur, against natural light, against neutral color, against dark color, against same color, partially occluded by hair, partially occluded by hand, partially occluded by body.

3.2 Camera Calibration

To account for the inevitable camera lens distortion, camera calibration was performed by calculating the intrinsic matrices and distortion coefficients, utilizing the checkerboard corner method from Zhang [99]. Optimal calibration was achieved by increasing the search window size and adding more examples from angles and far distances where periphery distortion was pronounced. In total, 25 varying images of the same checkerboard were used, and a window size of 40x40 pixels was applied when refining the corner locations. The height and width of each square were 25mm, and knowledge of this measurement was used to calibrate the pixel measurements to global coordinates in a metric unit - which was used later in the object measurement portion of the work. In testing different flags in the OpenCV [92] calibrate camera function, the best adjustment was to maintain the sixth radial distortion coefficient during optimization, while allowing the rest of the coefficients to change. Testing the fisheye

setting (commonly used in sport videos with the GoPro) resulted in a very high reprojection error, so it was disregarded.

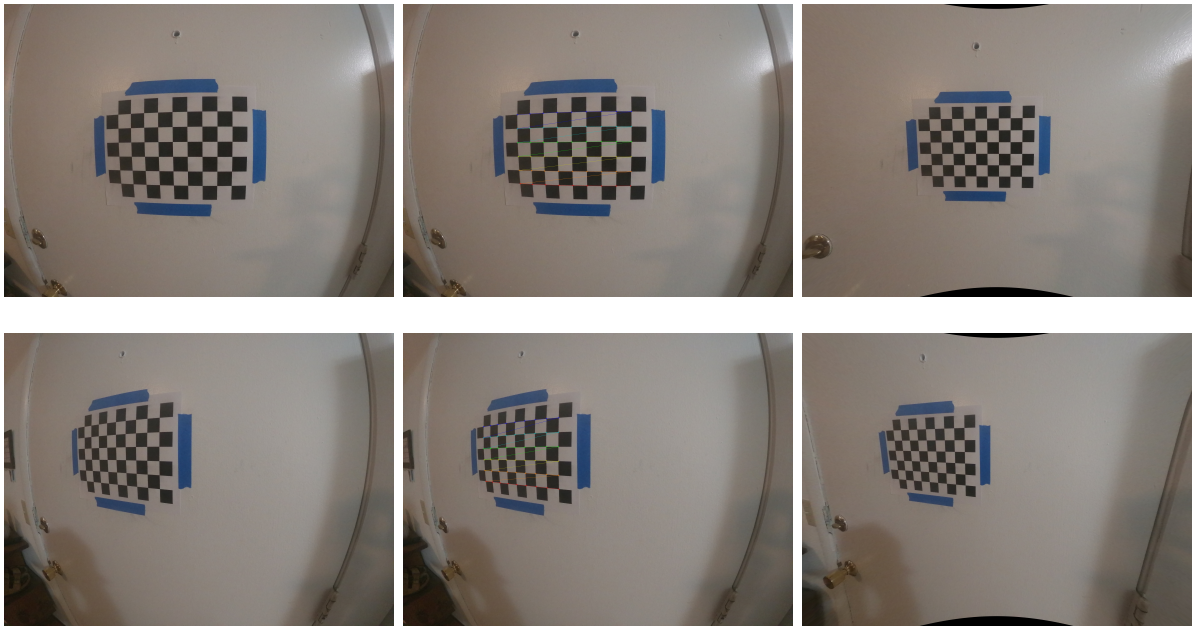


Figure 3.2: Two visual examples of the intrinsic camera calibration process - the first row at a straight forward angle; the second row at a sharp, peripheral angle. From left to right are the original image, the image with identified corners, and the corresponding undistorted image.

In the end, the total reprojection errors obtained were 0.0644 and 0.0700 for the left and right cameras, respectively. Before applying any tracking or measuring logic at runtime, the first step is always undistorting the current frame using the calculated calibration values.

3.3 Scene Calibration

Because the cameras are static throughout the entire throwing process and do not move between throws, some simple manual calibration can be done after the cameras are set up in the environment, but before the real time algorithm is initiated. The user is asked to view

a frame from the left camera and a frame from right camera and is prompted to click on the position of the center of the stop board, followed by its corner points, along with the corners of the throwing sector itself plus a few other corresponding points. Each image is divided into 25 squares, so the user finds a point correspondence in each square, ensuring the correspondences are scattered evenly throughout the images. These values will primarily be used for calibration between the two cameras, which is valuable for helping detect when the shot put begins its flight as well as the "8-point algorithm" for the measurement portion of the work.

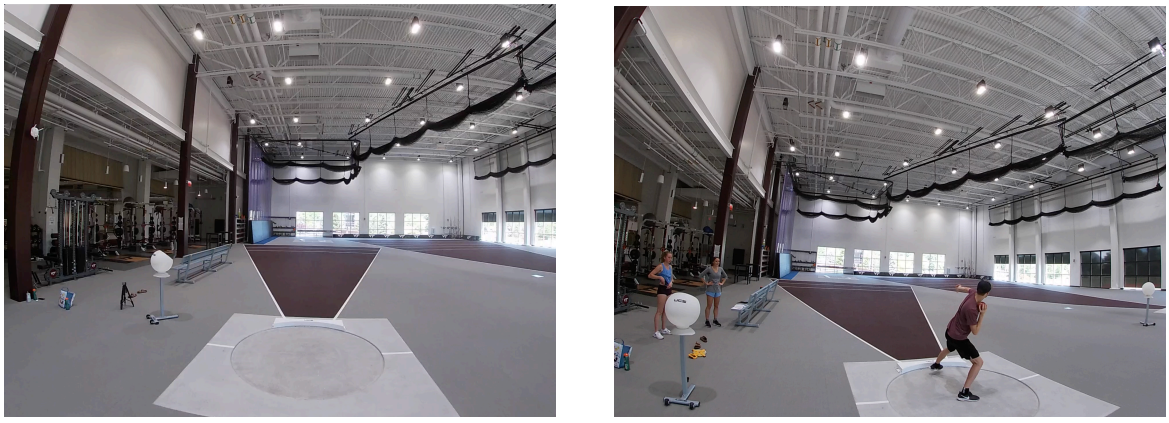


Figure 3.3: One example of an indoor shot put environment - the image on the left displays the area without any humans, while the image on the right displays the area when a thrower is in the shot put circle as well as a couple of spectators. Key areas are the throwing circle in which the thrower is standing, as well as the trapezoid-shaped throwing sector. If the thrower steps outside of the throwing ring during the throw, it is considered an illegal throw, and no measurement is recorded. The same result occurs if the shot put lands to the left or right of the throwing sector boundaries.

3.4 Object Tracking

To obtain an initial baseline, existing object trackers in OpenCV [92] were first tested to track the shot put - these include BOOSTING, MIL, KCF, CSRT, MedianFlow, TLD, GOTURN,

and MOSSE [106]. Most of these trackers performed well with objects large in size with respect to the image or with objects moving at a slower pace, but they generally failed to detect the shot put just a few frames after it left the thrower's hand. When waiting to initiate tracking until the shot put was in flight, the trackers still lost the object due to distractions from the surroundings. Some object trackers that apply optical flow, such as the Lucas-Kanade tracker [107] rely on keypoint detectors like the Shi-Tomasi detector [108] that looks for corners, and thus cannot track a round object. In this work, the object of interest appears to have several different shapes due to motion blur caused by fast movement, leading to less visual consistency across consecutive frames. *(See Discussion section for tracker results.)*

Instead of using an existing object tracker, I created a custom tracking heuristic specifically focused on one object of interest (shot put) that combines object detection with predictive path modeling that performs time-motion analysis and provides concurrent feedback. Object detection is the primary method for determining the shot put location in each frame, while path prediction serves an auxiliary role that both narrows the detection search window and provides a backup location if no reasonable detections can be found in a given frame.

3.4.1 Object Detection

Background Subtraction and Connected Components Filtering

Because the surrounding scene is relatively low-motion, especially compared to the high-motion, singular shot put throw, I first tried to identify connected components after background subtraction. The binarized image after subtraction *(see Figure 3.4)* allowed straightforward filtering through the OpenCV connected components algorithm [92]. Yet, it required a delicate balance of eliminating trivial noise, while maintaining detection of the shot put as it moved further from the camera and appeared smaller in each frame. For this reason,

the Gaussian blur applied to the current frame was incrementally decreased each iteration to raise the chance of detection, albeit with more noise. The MOG2 background subtractor [109] showed better resuts for a small object as opposed to the kNN subtractor [50], and was set to detect shadows to help with tracking the shot put in flight.

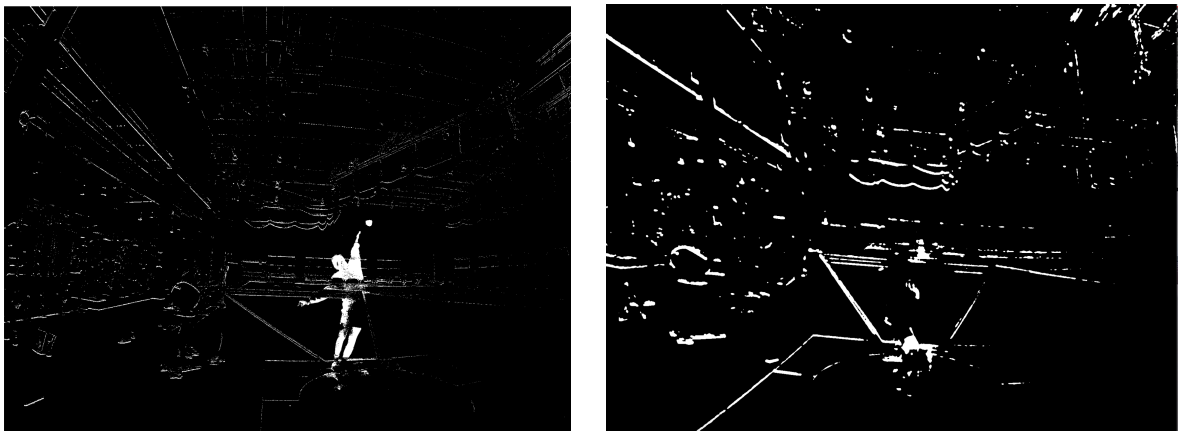


Figure 3.4: The binarized image as a result of the background subtractor, plus Otsu's thresholding as well as an added blur to eliminate futile noise. From the image on the left, one can see that the shot put is very clearly higlighted, with its round shape and relative size maintained. The thrower is clear as well, making it easy to identify the region of interest before flight of the shot put is initiated. However, on the right, one can see how unreliable background subtraction can be, especially in identifying small objects which are easily confused for noise.

Because the focus of this tracking mechanism is a pre-identified object, there is some generally known information that can be applied ahead of runtime to filter results from the connected components algorithm [110]. For example, the relative size of the shot put, and a percentage that it decreases per frame, as well as the ratio of width to height of the bounding box (close to 1 given the object is uniform and spherical). Figure 3.5 shows how the connected components' results were filtered down by pre-determined knowledge.

While this method worked well in the initial stages of tracking, as the shot put moved further away, it appeared as a miniscule object that was either disregarded as noise or blended so well that the background subtractor could not pick it up. Because of frequent blending,

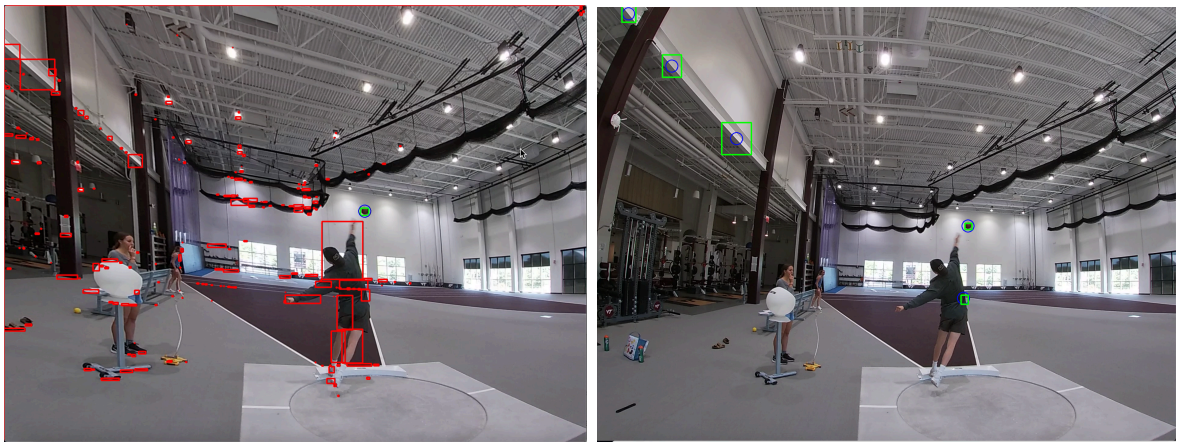


Figure 3.5: Example of the background subtraction, connected components and filtering heuristic that detects potential candidates for the shot put. While the number of candidates significantly decreased from filtering, it still made it difficult to detect the exact component of the shot put, leading to a high error rate. Because connected components is run on the binarized image, this approach also relies on a background subtractor that does not cancel out the shot put, which is not guaranteed.

color-based approaches are not viable solutions. Additionally, creating any type of color mask to identify the shot put would not be robust enough in all scenarios, since the shot put is not guaranteed to be a certain color. Figure 4.1 shows just how tiny the shot put is during time of flight, as well as how much the appearance of the object in two dimensions is affected by motion blur or background blending.

Even through adjusting blur and cropping the image towards a region of interest, the shot put still remained unidentified in multiple consecutive frames. This required too much reliance on extrapolation from path prediction, which after a certain number of frames, became too inaccurate to be the primary method of identifying the shot put location.

Deep Learning

Because non-learning methods were not accurate enough for detection, training a formidable deep learning model was the next step. The YOLO model was selected due to its real time capabilities as well as its common use in object detection problems in sports [64] [59]. Version 3 was chosen because it is more advanced than the original YOLO model, yet also more robust than the newest version 5 model. Initially, the YOLOv3 and YOLOv5 pre-trained models were downloaded and tested, since one of their 80 classes is a 'sports ball'. These results were worse than background subtraction for the object tracking portion once the ball was in flight, even when selecting a region of interest and upscaling the cropped image through pixel interpolation. Additionally, when only focusing on one or two specific objects, it was not worth the extra computation time to detect objects trivial to this specific application *(See Figure 3.6)*.

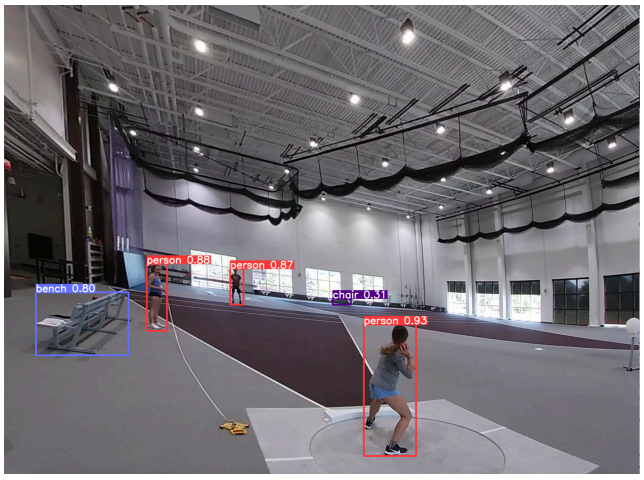


Figure 3.6: Result of running the YOLOv3 pre-trained model on an image, while it does well to identify people, including the thrower with fairly high confidence, it also identifies a chair, TV and bench, which are irrelevant in this system.

Object Detection for Flight Initiation

However, the YOLOv3 pre-trained model works well to detect when the shot put has started its flight, where the algorithm must identify that the shot put has just left the thrower's hand. The YOLOv3 'sports ball' class suffices to identify the shot put and the 'person' class can identify candidates for the thrower. Once the thrower enters the throwing ring, they have "60 seconds to release the shot put" [7] and in this time, the shot put needs to be detected when it has been released to give the tracker a starting point. While the tracker could theoretically be initiated at any time, it is likely both more accurate and efficient if it does not start until the path of the throw begins.

The current video frame is first run on the YOLOv3 detect.py script using the provided weights, and a list of detections is returned. The list is scanned for at least one sports ball detection and at least one person detection. If either is not found, the script then repeats on the subsequent frame. If both detections are found, a search window is constructed based on the stop board and throwing sector coordinates obtained during scene calibration. If the frame originates from the left camera, the search window is adjusted to the right and vice versa if the frame originates from the right camera. This is based on the knowledge that the center of the throwing ring is skewed to the right in the left camera and skewed to the left in the right camera. *(See Figure 3.7).*

The detections are filtered within the search window and again, if either is not found, the script moves to the subsequent frame. If there is at least one sports ball and at least one person detection inside the search window, there is further criteria to ensure the thrower has released the shot put. For each sports ball detection within the search window (which more than likely will only be one), the closest person detection is found, and the two bounding boxes are compared. If the bounding box of the shot put is completely nested within or

Figure 3.7: Images of the constructed search window based on coordinates of the stop board and throwing sector in each frame obtained during scene calibration. The width of the box is adjusted by a factor of 20% of the frame width to the left or right from the stop board coordinates, and the height is increased by 20% of the frame height from the sector coordinates.

overlaps the bounding box of the person by more than 60%, it is considered to still be in the hands of the thrower, therefore is not deemed in flight. If the bounding box of the shot put is too far away from the bounding box of the person, then the person is most likely not the thrower and the shot put is not in flight.

Therefore, the final criteria is that if there is at least one combination of sports ball and person within the search window that are less than a set distance away from each other and do not substantially overlap, then it can be reasonably concluded that the shot put has been released from the thrower's hands and flight has been initiated. The shot put coordinates are then sent to the tracking script, to follow the path of the shot put and ultimately detect the coordinates of the landing point.

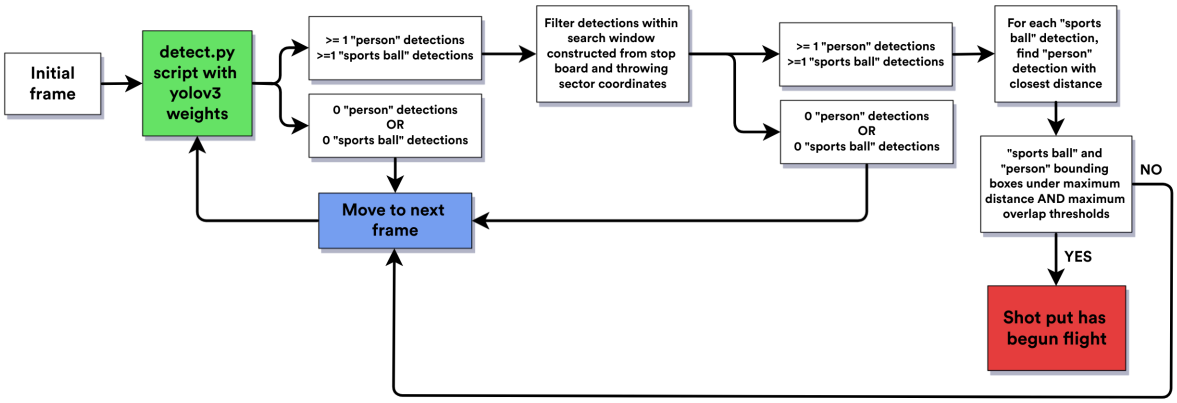


Figure 3.8: Workflow of the logic applied to determine when the shot put has left the thrower's hand and begun its flight based on object detection results from a pre-trained deep learning model and information known ahead of time. Afterwards, the coordinates of the shot put are sent to the tracker, which is then initiated.

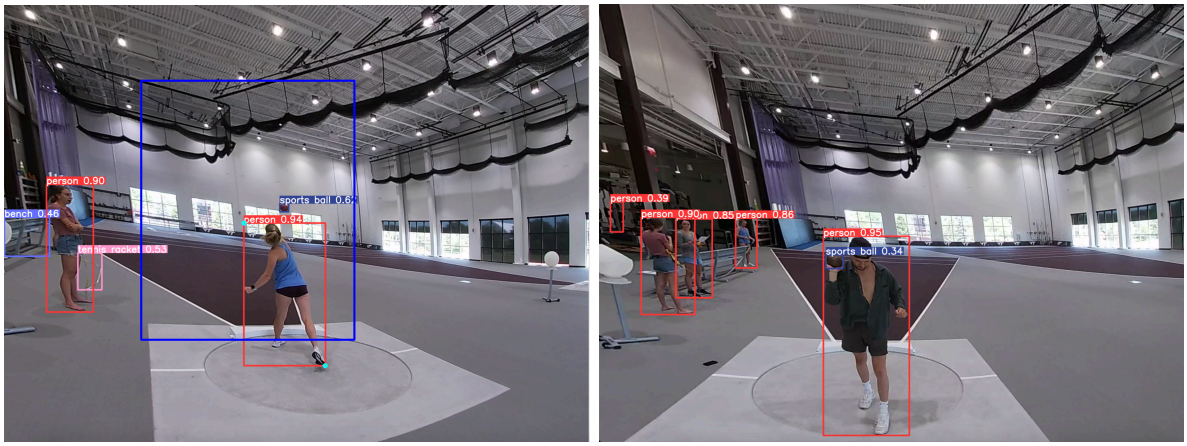


Figure 3.9: Examples of running the YOLOv3 pre-trained model on a frame to detect if the shot put has been released from the thrower's hand. The left image shows a scenario when the shot put is in flight where the center of the thrower and shot put are within the blue search window and do not overlap, but fall within the distance threshold. The right shows where the algorithm knows the shot put is not in flight because the bounding boxes are nested, meaning there is too much overlap. The YOLOv3 model often detected the shot put when it was still in the hand of the thrower, so extra logic was applied to ensure this did not result in a false positive "in flight" result.

Object Detection in Flight

Because built-in object trackers and pre-trained deep learning models were not accurate enough for the tracking portion of my application, to increase the ratio of successfully detected frames, I trained YOLOv3 [61] through Ultralytics [111]. It was independently trained on three custom datasets, then the best model was selected.

In training, rather than randomizing the weights, the given YOLOv3 weights were used as the initial values. The batch size was set to 8 and the image size was set to 640, as these were the largest sizes the CPU could handle. Each model was set to train for 2000 epochs, based on the recommended number for one object, but each stopped at an earlier epoch when improvements leveled out. The training script was run on a Nvidia GeForce RTX 2080 Rev. A GPU in a machine with an Intel Xeon i7 5500 CPU running at 3.3 GHz and 96 GB of RAM.

The 'original' dataset consists of 2100 images, originating from 46 videos, where priority is given to frames where the shot put is in flight as opposed to videos where it is stationary. The shot put was manually labeled in each frame, through the Labelbox API [112]. This model trained for 456 epochs, which took 15.147 hours.

While the amount of data collected was sufficient, it was not thorough enough for the level required to train deep convolutional neural networks. Additionally, there was a need for more variability in the data to forestall overfitting. To account for these potential issues, data augmentation was performed on the existing frames in order to diversify the dataset and achieve better training results. Alternative strategies of copy-pasting the object of interest [70] to increase its frequency in the dataset were rejected due to their likelihood to overfit since the images would be obviously synthesized.

Using the Albumentations library API [76], ten transform augmentations were applied on

each frame from the original dataset, and together, all augmented and original frames were used to train as the second model. Example images of augmentations are provided in Figure 3.10. Color jitter (which randomly adjusts brightness, contrast, saturation and hue), as well as random shadow, random sun flare and random tone curve account for possibilities of differences in the environment. Horizontal flip, vertical flip, and diagonal flip were chosen to help learn different orientations of the object. Gaussian blur, Gaussian noise, and motion blur were implemented to account for a moving object, especially in less precise settings. In total, this dataset is comprised of 23,101 labeled images - a ten-fold increase to the original dataset. Bounding box coordinates of labels were programmatically adjusted if the position of the shot put was changed due to the augmentation.

For the third dataset, true negative labels were added to the original dataset, with the idea that if the model learns objects that are commonly confused for the shot put, the number of false positives will decrease. The three additional labels were "light", "weight", and "other", which included various small objects in the environment like fire alarms and water bottles.

The test set consisted of 1,086 manually labeled frames [112] from 8 throwing videos that were held out from the training set. All models were tested on the same frames to compare results. Quantitatively and qualitatively, the second model - the augmented model - demonstrated the best results, so its most recent weights were chosen (rather than best weights to avoid overfitting). The weights were run with the YOLOv3 detect.py script on each frame of the video with a confidence threshold of 0.25. *(See Discussion section for analysis of the three training models.)*

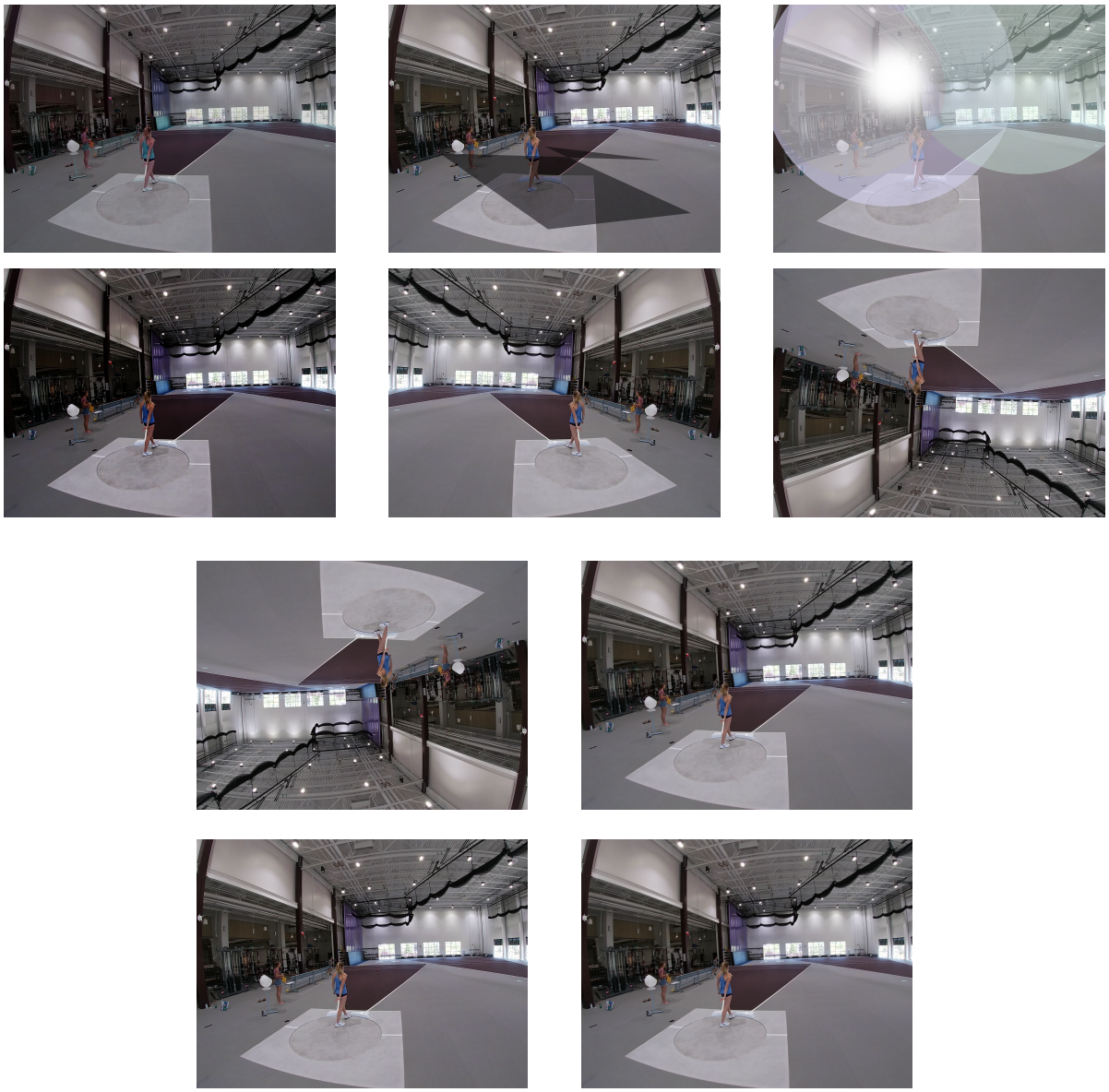


Figure 3.10: A display of the ten different augmentations performed, all originating from the same frame. From left to right, top to bottom are color jitter, random shadow, random sun flare, random tone curve, horizontal flip, vertical flip, diagonal flip, Gaussian blur, Gaussian noise, and motion blur.

3.4.2 Path Predictive Modeling

The YOLOv3 detect.py script is slightly modified to return a list of coordinates that denotes the pixel location of all candidates for the shot put in the given frame, as well as the confidence value. From there, the best candidate must be identified, followed by a decision to determine if this candidate is a valid location for the shot put. First, the candidate with the shortest Euclidean distance from the previous shot put location is selected and then, if the candidate's location falls within a pre-determined search window based on the previous location, it is recorded as the shot put coordinate for the given frame.

To eliminate the chances of selecting a false positive detection as the shot put coordinates, the search window is adjusted based on two criteria:

1.) By a ratio to the area of the bounding box - meaning the larger the shot put in the previous frame, the larger the search window in the current frame. As the object becomes smaller, there is less need for the search window to be large, therefore it incrementally decreases.

2.) By a factor of number of frames since the last valid detection - meaning, the longer it has been since the shot put candidate was detected by the custom model, the greater the search window. This helps account for any error due to repeated extrapolation.

Deep learning is not infallible and in the event the object detector returns zero detections or the closest detected point fails to meet the search window plus confidence score criteria, the predicted location is recorded as the shot put coordinates in the given frame. The predicted point is extrapolated through multiple kinematics formulas based on previous position, velocity and acceleration measurements. Figure 3.11 shows the workflow at each frame to construct a sufficient path for the shot put throw from a given video.

It is important to note that these point and path predictions are maintained in two dimensions, where the values are measured by changes in pixels per frame, rather than a three-dimensional, global coordinate system. Because of this design decision, even though a shot put throw travels in a projectile motion, typical three-dimensional projectile formulas cannot be used, such as physics formulas assuming a known acceleration due to gravity (9.8 m/s^2). In addition, due to non-negligible factors that affect trajectory such as air resistance or rotation of the implement, a more accurate path is generated by performing instantaneous predictions at each frame.

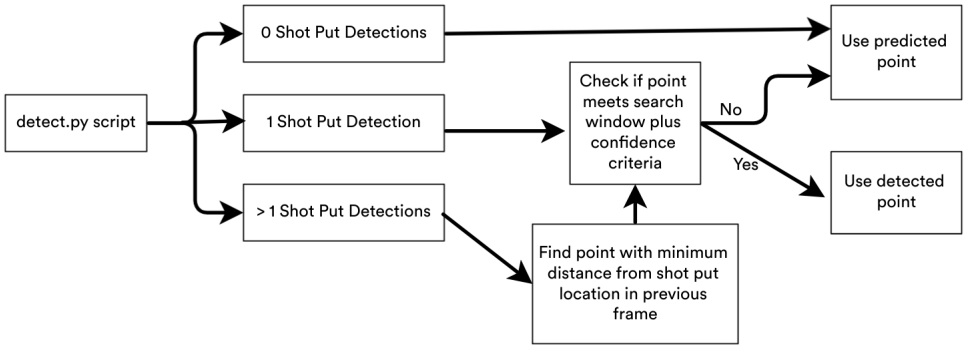


Figure 3.11: Workflow of path prediction and construction in each frame. Initially, the detect script is called on a frame, and returns any number of coordinates for candidates for the shot put. Different steps follow depending on the number of detections, but all lead to either recording the detected point or the predicted point for the coordinates of the shot put in the current frame.

General kinematic formulas to calculate change in position relate the three key variables of motion - position, velocity and acceleration. They assume constant acceleration and are only concerned with one direction, therefore basic integration can be applied. The formula below represents the ideal case where the definite integral can be calculated; however, this is rare in the real world due to non-trivial factors that affect the path as well as a lack of run-time knowledge of some crucial values.

$$\Delta P = \int_0^t (V_{\mathrm{i}} + at)\, dt$$

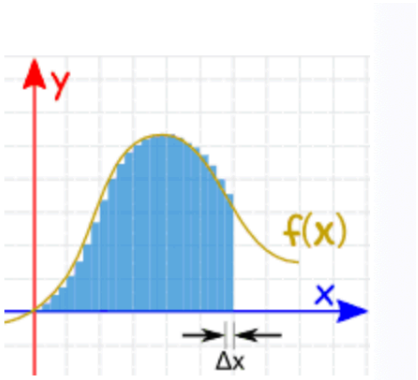
$$\Delta P = (V_{\mathrm{i}} t + \frac{1}{2} a t^2)\Big|_0^t$$

$$\Delta P = V_{\mathrm{i}} t + \frac{1}{2} a t^2$$

ΔP = change in position. V_i = initial velocity, a = acceleration, t = time

In this work, since both x and y positions are changing at different rates and acceleration cannot be assumed as a constant in either direction, the above formula must be adapted to incrementally calculate change in position, velocity, and acceleration for both directions independently. The change in position in the x-direction is the change in pixel column-value across frames, whereas the change in position in the y-direction is the change in pixel row-value. In all equations, n represents the current frame and *n-1* refers to the previous frame.

While acceleration is not constant in the x-direction, nor 9.8 m/s^2 in the y-direction, it is still acceleration due to gravity, not due to any forces related to the shot put. A time step is a frame in this application, which is equivalent to $\frac{1}{30}$ seconds, because the cameras advance at 30 FPS. The above integral is instead approximated by summing the change in position at each frame when the shot put is in flight *(See Figure 3.12 for a summation approximation visualization).* Below shows how the integral for change in position between two frames translates to the predicted point calculation.



$$Displacement \approx \sum_{i=1}^{N} f_{\text{i}} \Delta x$$

Figure 3.12: A two-dimensional graph that shows how the summation approximation method can be a sufficient substitution for an exact integral calculation - each rectangle is equivalent to a frame, which is equal to $\frac{1}{30}$ second. In this specific application, displacement is the distance the shot put has traveled, which is approximately the summation of the change in position at each frame. $i = 1$ represents not the initial frame of the video, but the first frame where the shot put is detected to be in flight. N represents the frame where the shot put makes initial contact with the ground. *Image Credit: [113]*

$$\Delta P = P_{\text{n}} - P_{\text{n-1}} = \int_0^1 (V_{\text{n-1}} + a_{\text{n-1}} f)\, df$$

$$P_{\text{n}} = P_{\text{n-1}} + \int_0^1 (V_{\text{n-1}} + a_{\text{n-1}} f)\, df$$

$$P_{\text{n}} = P_{\text{n-1}} + (V_{\text{n-1}} f + \frac{1}{2} a_{\text{n-1}} f^2)\Big|_0^1$$

$$P_{\text{n}} = P_{\text{n-1}} + (V_{\text{n-1}}(1) + \frac{1}{2} a_{\text{n-1}}(1)^2) - (V_{\text{n-1}}(0) + \frac{1}{2} a_{\text{n-1}}(0)^2)$$

$$P_{\text{n}} = P_{\text{n-1}} + V_{\text{n-1}} + \frac{1}{2} a_{\text{n-1}}$$

P = position, f = frame, df = change in frame

The above approximation for change in position is also a version of the 2nd degree Taylor polynomial, known as a quadratic approximation, since position is twice-differentiable. The first equation is the general format of the quadratic approximation. The following lines demonstrate how it is applied to estimate position in the current frame based on the first derivative of velocity and the second derivative of acceleration from the previous frame:

$$P(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f"(a)(x-a)^2$$

x = n = current frame *a = n – 1 = previous frame* *P(x) = position(x)*

f(a) = position(a) *f'(a) = velocity(a)* *f"(a) = acceleration(a)*

$$\boldsymbol{P(n) = position(n\text{-}1) + velocity(n\text{-}1)*1 + \frac{1}{2}acceleration(n\text{-}1)*1^2}$$

The velocity and acceleration are also constantly updated in a feedback loop to improve accuracy of the approximation - based on the knowledge that velocity is the derivative of position, and acceleration is the derivative of velocity:

$$V_{x_n} = P_{x_{n-1}} - P_{x_{n-2}} \qquad V_{y_n} = P_{y_{n-1}} - P_{y_{n-2}}$$

$$A_{x_n} = V_{x_{n-1}} - V_{x_{n-2}} \qquad A_{y_n} = V_{y_{n-1}} - V_{y_{n-2}}$$

To prevent the predicted points from causing the constructed path to be too far from the true path, once the shot put is detected again after relying on predicted points for multiple frames, the velocity and acceleration values are recalculated in both directions. The equation is based on the average velocity formula for change in position, and uses the current and most prior detected coordinates as well as the number of time steps between the two frames:

$$\Delta P = \frac{(V_{\text{i}} + V_{\text{f}})}{2} * time$$

$$P_{\text{current}} - P_{\text{prior}} = \frac{(V_{\text{prior}} + V_{\text{current}})}{2} * (current - prior)$$

$$V_{\text{current}} = 2 * \frac{P_{\text{current}} - P_{\text{prior}}}{(current - prior)} - V_{\text{prior}}$$

$$current = current\ frame\ number,\ prior = prior\ frame\ number$$

In the case of the first and second frames when the shot put is in flight, velocity and acceleration cannot yet be calculated, therefore the predicted position is equal to the detected position. In the third frame, the predicted position accounts for velocity, but still not acceleration. Once the path contains three points, then both acceleration and velocity can be factored in to the prediction formula for the shot put.

The ability to measure instantaneously and constantly update these motion values at each frame is a major benefit of real time detection. This allows a better predicted path of the shot put because it accounts for the effect of any perturbations caused by outside forces, rather than relying on projectile formulas with too many assumptions. It also provides the ability to report statistics in real time, such as angle and average velocity of the shot put that may be beneficial to users of the system. Appendix A.2 provides a dictionary of all values calculated and stored at every frame when the shot put is in flight. In the end, a path is constructed based on a precisely calculated fusion of the detector and predictor to ensure the most crucial point - the landing point - is accurately detected.

3.4.3 Landing Point

The most important point of the entire shot put throw is where it makes initial contact with the ground because this serves as the official point of measurement. Irrespective of how far the shot put travels or how many times it bounces, as long it lands within the bounds, it is always measured from the that initial contact point. In order to determine this point, the velocity in the y-direction is tracked to find the frame where the velocity vector changes direction. There are two key frames that contribute to the landing point - the frames just before and just after the shot put hits the ground because the direction of the y-velocity vector changes, meaning something must have exerted force on it. Therefore, it can be concluded the shot put has landed on the ground and bounced. To then predict the exact pixel coordinates where the shot put landed, only the detected coordinates from the last frame before the shot put makes contact with the ground are considered, since they are the last known coordinates during flight.

The landing point is interpolated by calculating the time step to get from the current velocity of the last frame to a velocity of exactly 0, known as a critical point since the derivative of position is 0. Once the time step is calculated, it is inserted into the predicted position formula to find the predicted x and y coordinates of the actual point of landing. It is assumed in the calculation that outside forces, such as lift and drag trivially affect the shot put's flight because it is so close to the ground in the last frame.

$$V_{\text{final}} = V_{\text{initial}} + a * time$$

$$V_{\text{landing}} = V_{\text{current}} + a * time\_step$$

$$time\_step = \frac{V_{\text{landing}} - V_{\text{current}}}{a_{\text{current}}}$$

$$P_{\text{predicted}} = P_{\text{current}} + V_{\text{current}} * time\_step + \frac{1}{2} a_{\text{current}} * time\_step^2$$

An advantage of tracking the landing point in the left and right cameras independently is that any errors due to lack of synchronization between the two cameras are accounted for. For example, the landing point might be detected in frame number 57 for the left camera and frame number 60 for the right camera - they do not have to be detected in the same frame number.

Predictions made during object tracking must be robust enough to handle any range of launch angle and distance thrown. There is a wide range of trajectories, especially at the beginner levels, thus the algorithm must be equipped to track the path and estimate landing point regardless of the shape of the trajectory, which is yet another reason why instantaneous measurements are advantageous.

Figure 3.13: Detected landing point of the shot put after tracking the path. Predicted points are in pixel coordinates. Left and right frames both shown.

3.5 Object Measurement

The fundamental matrix to triangulation pipeline based off mathematical axioms from Hartley and Zisserman [87] is the ideal method for reconstruction because it does not require any

predetermined information about the relationship between cameras, nor does it require cameras to be perfectly parallel to one another. These principles greatly reduce the amount of potential human error when computing rotation and translation between cameras.

One major benefit of a system with stationary cameras as opposed to moving cameras is that several values only need to be calculated once during set up, as these values will remain unchanged throughout run time. Calculations include estimating the essential matrix, determining the projection matrices, and triangulating the global point for the stop board. The only aspects of measurement that cannot occur until run time are triangulating the global point of the shot put's landing point and calculating a final distance of the throw.

Because camera calibration has already been performed, intrinsic matrices and distortion coefficients are known, meaning the essential matrix can be calculated from the mathematical relationship between the fundamental matrix and intrinsic matrices. The set of given points is compiled from a user who manually clicks corresponding points in the two images during set up. Obtaining accurate point correspondences is crucial to the success of measurement because the rest of the operations stem from the essential matrix and testing throughout relies on the validity of the known points. Even though only eight points are required for the 8-point algorithm [88], they must be as precise as possible, which becomes more difficult as both the number of pixels of an image and the warping between images increase.

Therefore, more than eight sets of matching points are clicked, but further than that, each set of points undergoes a refining process to improve the correspondence accuracy. For each set of points, template matching is performed using a template window constructed around the left point of dimensions of 10x10 pixels, and a search window around the right point of 50x50 pixels. The result from OpenCV's matchTemplate() function is then upscaled by a factor of eight by resizing with cubic interpolation. The location of the maximum value is found in the upscaled image, which is then downsized to return the point in terms of the

original image - this process leads to sub-pixel accuracy approaching an eighth of a pixel. The same process repeats, using the refined coordinate in the right image to construct a template window and then constructing a corresponding search window in the left image. This increases the accuracy between point correspondences in each image.

The refined matching points are then sent to the OpenCV findFundamentalMat() function, which returns the fundamental matrix, using the best eight point correspondences found through the RANSAC method [114]. From there, the essential matrix is calculated by multiplying the intrinsic matrices of the left and right cameras by the fundamental matrix:

$$E = K_{\text{left}} * F * K_{\text{right}}$$

where E is essential matrix, K is intrinsic matrices and F is fundamental matrix

The essential matrix is verified by calculating the average error of the normalized known points, which should be extremely close to zero. Once there is enough confidence in the essential matrix, it is decomposed into two rotation matrices and two translation vectors through Singular Value Decomposition (SVD). Four possible projection matrices for the right camera are created by concatenating the resulting rotation and translation solutions and multiplying them by the right intrinsic matrix. Based on the Hartley-Zisserman steps, the left camera will be the reference frame for all coordinates, therefore the left projection matrix is simply its intrinsic matrix multiplied by the identity matrix concatenated with a column of zeroes. *See equations below where K denotes the intrinsic matrices, P the projection matrices, R the rotation matrices, and t the translation vectors.*

R1, R2, t1 = cv2.decomposeEssentialMat(E)

$$t2 = -t1$$

$$P_{left} = K_{left} * [I|0]$$
$$P_{right1} = K_{right} * [R1 \mid t1]$$
$$P_{right2} = K_{right} * [R2 \mid t1]$$
$$P_{right3} = K_{right} * [R1 \mid t2]$$
$$P_{right4} = K_{right} * [R2 \mid t2]$$

The correct projection matrix is found by triangulating all of the known corresponding points four times each - one time for each possible right projection matrix. Triangulation is based on finding best fit using least squares so that backprojection is as accurate as possible in both the left and right images, as it will never be perfect due to noise. The relationships between the pixel coordinate, global coordinate and projection matrix are converted to a linear system, where the cross product between the pixel coordinate and the result of the projection matrix multiplied by the global coordinate equals zero. The homogeneous linear system is then solved through SVD to find the best global coordinate from the given two pixel coordinates and two projection matrices [115].

The global points that result from triangulation are tallied based on positive depth, which is a requirement since all points should be in front of the camera. Next, the four projection matrices and presumed global points are used to perform backprojection, which results in pixel coordinates for the left and right images. The pixel coordinates will be homogeneous, therefore will also produce a depth value that should be positive. The correct right projection matrix is selected as the projection matrix that produces the maximum number of points with positive depth, in both pixel and global coordinates. *(See Figure 3.14).*

Once the correct projection matrix for the right camera is found, the global coordinates of the stop board are triangulated, which will remain the same throughout the throws.

Algorithm 1 Find correct projection matrix

```
for possible projection matrix do
  for known pair of point correspondences do
    g ← triangulate(globalpoint)
    if depth(g) > 0 then
      count(projectionmatrix) + +
    end if
    leftpoint ← backproject(leftimagepoint)
    if depth(leftpoint) > 0 then
      count(projectionmatrix) + +
    end if
    rightpoint ← backproject(rightpoint)
    if depth(rightpoint) > 0 then
      count(projectionmatrix) + +
    end if
  end for
end for
best projection matrix ← maxcount(projectionmatrices)
```

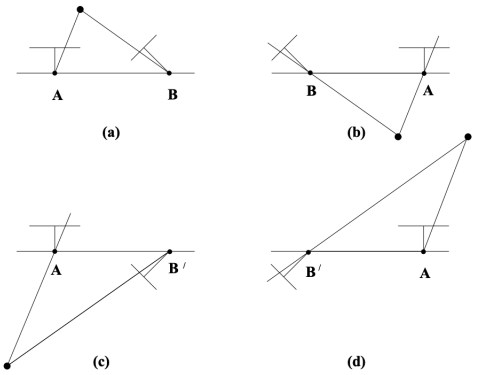


Figure 3.14: Visualization of triangulating the global point and performing backprojection to obtain image points - **(a)** is the correct solution because it is the only solution with both global and pixel coordinates in front of the cameras (i.e. with positive depth). *Image credit: [88]*

Finally, during runtime, after the coordinates of the shot put landing are identified for both cameras, the shot put global coordinate is triangulated using the right projection matrix. To account for the scale factor and convert from pixel to metric units, all global coordinates are multiplied by 1 / focal length [100], which is 1 / 0.033 meters for the GoPro cameras

used in this study. Then, the Euclidean distance between the shot put and stop board global coordinates are calculated; however, only the x (width) and z (depth) coordinates are used, since shot put measurement rules [7] disregard any height differences of the two points.

$$Distance = \sqrt{(x_{\text{stopboard}} - x_{\text{shotput}})^2 + (z_{\text{stopboard}} - z_{\text{shotput}})^2}$$

Chapter 4

Discussion

4.1 Experimental Results

4.1.1 Training Custom YOLO Models for a Singular, Small Object

Training the custom YOLO models led to much better performance than the pre-trained model. I hypothesize this is because many object detection algorithms, like YOLO, focus on accurately detecting as many objects as possible, rather than one or a few specific objects. Most pre-trained object detection models also tend to reduce image resolution, meaning small objects often disappear in the middle of the network, thus never reaching the detection phase [61]. One can see in Figure 4.1 just how small the shot put is during tracking, thus how important it is that the model is able to detect small objects.

Consistent with published results on data augmentation, augmenting the data tremendously improved training results on the YOLOv3 framework. Even though training time was considerably longer, it yielded much higher mean average precision (mAP) values, both @0.5 and @[0.5:0.95]. The reason results were better is likely due to the greater variation of shot put examples - in Figure 4.2, one can see how augmenting the data through simple pixel augmentations significantly increases the diversity of location of the shot put. The mAP and F1-scores are used as metrics to quantify the efficacy of each model. Each model reached a high mAP @0.5 quickly; however, because the focus is as high an accuracy as possible

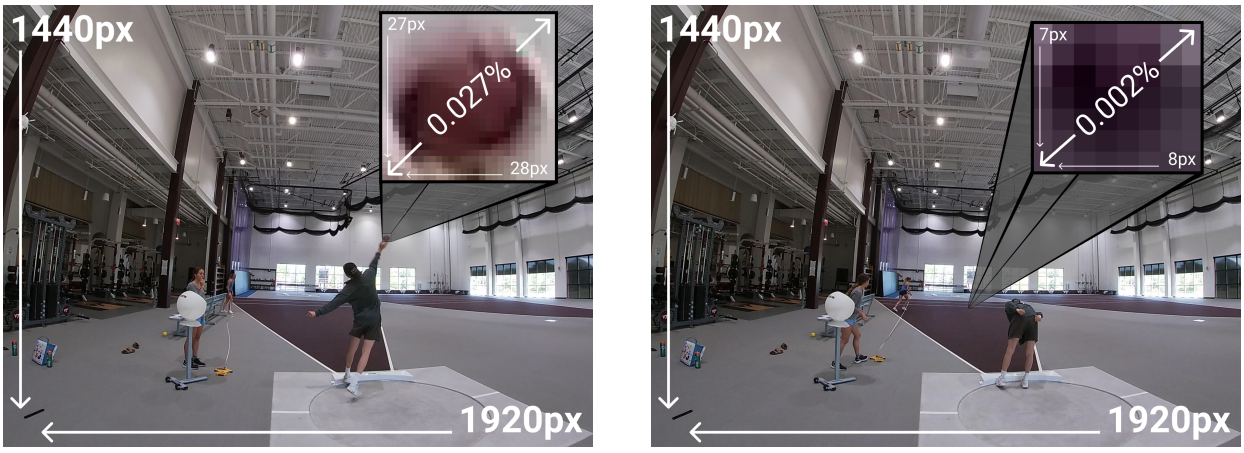


Figure 4.1: A visual demonstration of the small size of the shot put with respect to the size of the image. The left image shows when the shot put is closest to the camera right before it leaves the thrower's hand - falling into the definition of a small object and representing only a small percentage of the total image area. It also shows the effect of motion blur, which compounds the difficulties. The right image shows when the shot put is furthest away from the camera - just as it is landing on the ground. The dimensions of the shot put in this image match the tiny object definition. This image also shows how difficult it can be to distinguish the object of interest from its background - almost imperceptible even to the human eye. While the images are of high quality, the percentages of the object size relative to the image are a mere 0.027% and 0.002% for the left and right images, respectively.

for one specific object of interest, I focus on mAP@[0.5:0.95], whose results are shown in Figure 4.3. Not only does the original dataset level out at 0.6, but it also experiences some dramatic fluctuations. Both the augmented and negative datasets reach significantly higher mAP values, but ultimately the mAP for the augmented dataset is higher. The F1-scores follow the same pattern *(See Figure 4.4)*, yet all three datasets do reach a high F1-score with both high precision and high recall. This implies a reliable model with few false positives and false negatives. When running the detector, there was usually only a maximum of 3 shot put detections in a given frame, supporting the claim that there are few false positives.

Even though the results are significantly better after training with augmented data, there is a tradeoff in training time, which is more than 30 times longer *(See Figure 4.5)*. Based on the results, it is worth a longer training time, which is still a reasonable amount of time in

retrospect (14 days compared to less than 1 day).

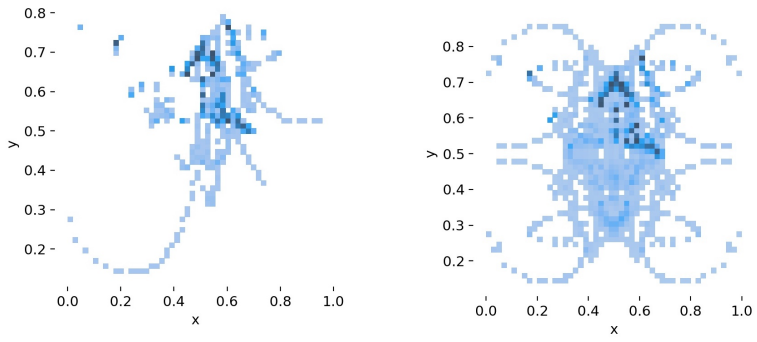


Figure 4.2: Contrast of the position diversity of original versus augmented datasets. The augmented dataset covers a much wider range of positions within the frame, increasing the likelihood of accurate shot put detection.

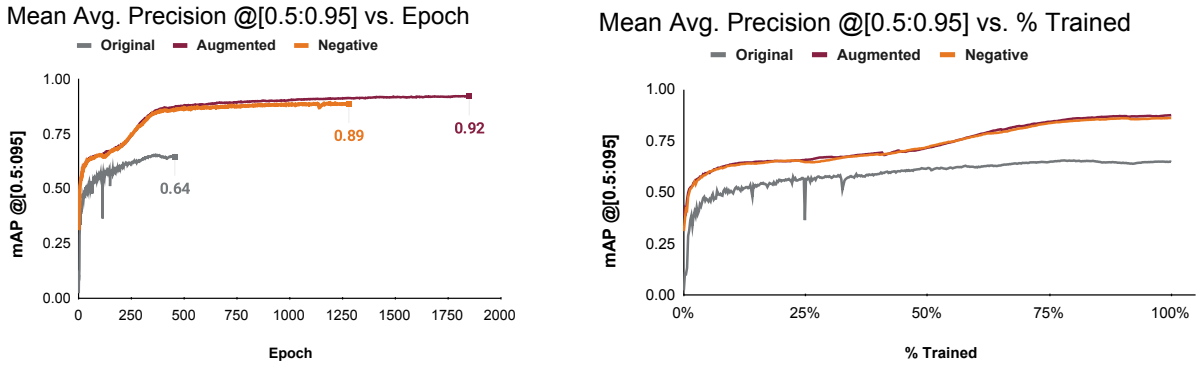


Figure 4.3: f
or each model.]Display of the mAP values @[0.5:0.95] for each model. The left chart displays the progression through each epoch - the right chart displays the same data, but the data is rescaled in the form of percentage to study each. The data point labeled for each dataset is the final mAP value calculated for each.

Out of the three less than ideal scenarios for detecting the object of interest, the small object problem and motion blur problem were both solved easily through training a CNN. By far, the most evident problems regarding obscure scenarios were cases where the shot put blended in with its background. Custom training on YOLOv3 led to increased identification of small objects and moving objects, as long as the background was distinct from the object.

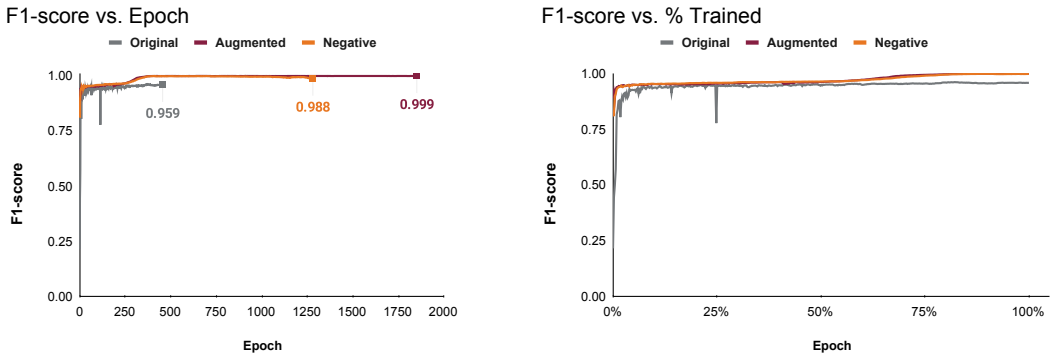


Figure 4.4: Display of the F1-score for each model, which is calculated from the precision and recall values at each epoch. The left chart displays the progression through each epoch - the right chart displays the same data, but rescaled in the form of percentage to better compare the trend in values for each model. The data point labeled for each dataset is the final F1-score calculated for each.

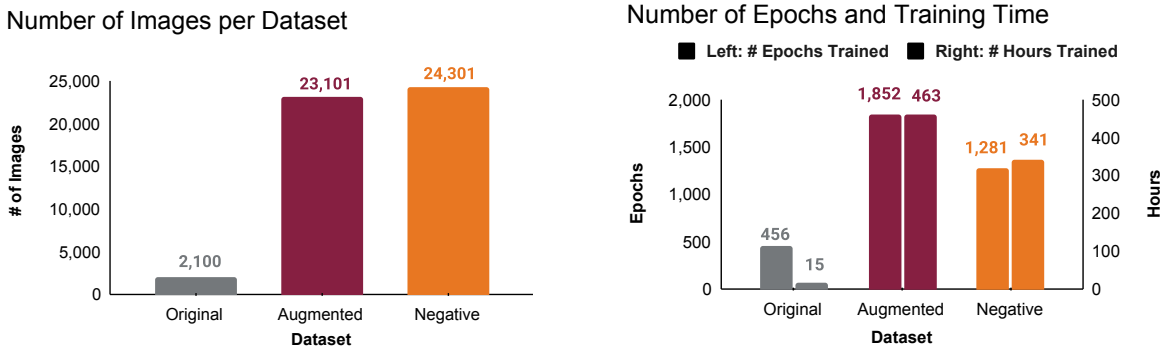


Figure 4.5: Visual demonstration of the difference in metrics for each dataset. The left shows the number of images in each dataset inputted for training. The right graph shows the difference in number of epochs and total number of training hours for each dataset.

Instances where the shot put blended with dark backgrounds were the most difficult for the detector to identify. Interestingly, it was still able to detect the shot put when it landed on the ground, which was a very similar color to the shot put.

4.1.2 Power of Predictive Path Modeling

The custom YOLOv3 model can detect many different instances of the shot put; however, it is never perfect. Identifying the wrong object as the shot put can greatly affect whether or not the correct landing point is recorded, therefore the combination of path prediction plus object detection proved very effective. Path prediction also serves as a valuable backup system when no detections are returned in obscure cases that even a powerful CNN model cannot learn, such as background blending. For example, Figure 4.6 shows all of the YOLOv3 detections for the shot put throughout the throw.

Figure 4.6: The difference between only running the detector (left image) and running the detector plus predictor (right image) throughout the shot put throw. The circles in green originate from the detector, whereas the circles in yellow originate from the predictor. One can see how adding the predictor fills in the path when there is background blending, even if it veers slightly off course. It also helps filter out the false positives, such as the thrower's hand and other round objects in the frame.

4.1.3 Measurement Results

Measurement accuracy is calculated as a percent error of the difference of true distance and calculated distance divided by the true distance multiplied by 100. Testing occurs on 11

different videos, but videos with the same set up (same height and baseline) use the same point correspondences, which imitates the idea that clicking point correspondences would only be a one time occurrence during each competition. Several combinations of parameters were tested to find which resulted in the lowest average error. *(See Figure 4.14).* While error is relatively high in some instances, there were several outliers contributing to error, thus the issue may lie more in consistency, rather than accuracy. For example, in the scenario with the best error value of 201%, when removing one outlier, the error decreases all the way to 82%. When running the algorithm on my personal machine, the YOLOv3 pre-trained model to detect flight initiation runs at an average of 3.74 seconds per frame, while the custom-trained model to track flight takes 2.99 seconds per frame. On average, from the time the shot put leaves the thrower's hand to the time a calculated throw measurement is returned is an average of 4.5 minutes. *(See Figure 4.7).* The best measurement results from a combination of 9 manual point correspondences as opposed to SIFT, a scale factor of 8, a search window of 20x20 pixels, template window of 16x16 pixels, beginning with the fundamental matrix rather than straight to essential, and refining coordinates in the left and right images.

Each global coordinate is returned as a three-dimensional coordinate in the order width, height, depth. The coordinates are directionally accurate, meaning all values are positive since they are in front of the camera, the height values are fairly close together, and the depth value of the shot put is greater than the depth value of the stop board.

4.1.4 Error Optimization for Measurement Portion

Several techniques and parameters were tested to find the best and most consistent results of measurement. As discussed in Chapter 2, three techniques were tested - the Hartley-

	True Distance (m)	Measured Distance (m)	Error (%)	Computation Time (mins)
	10.02	7.23	27.89	4.23
	11.04	2.04	81.49	4.89
	12.50	8.32	33.44	4.21
	6.47	96.36	1,389.34	4.67
	12.10	20.92	72.93	5.10
	7.79	27.69	255.50	4.31
	5.78	3.77	34.70	4.08
	11.08	3.04	72.56	4.33
	4.07	7.71	89.43	3.98
	6.36	9.21	44.81	4.55
	8.21	17.42	112.18	4.78
Average:	8.67	18.52	201.30	4.47
Average w/o Outlier:	8.90	10.74	**82.49**	**4.45**

Figure 4.7: Table of final results using test videos and the best combination of input factors from the pivot table. Highlighted in orange is the single outlier that significantly skews the data - the average error and computation times are calculated both with and without this outlier. In bold are the final average error and computation time after subtracting the outlier.

Zisserman method, the depth map method, and rectification method. The depth map method through the OpenCV StereoBinarySGBM class required a great deal of trial and error and refining to obtain a suboptimal depth map. Additionally, this method was not the best use of time and resources given that the depth of most coordinates is not needed for the final result of this application.

The rectification method was better for only focusing on the important points of interest; however, the image adjustments were not always the most accurate, resulting in the y-axis values differing by around 20 pixels. Moreover, rectification error on cameras with a small baseline ran the risk of returning a negative disparity value, which invalidates the measurement result. The method also relies on a baseline input, which is measured by humans and a slight error in this value could compound to a greater error in the final measurement of the shot put throw. While the Hartley-Zisserman method struggles with

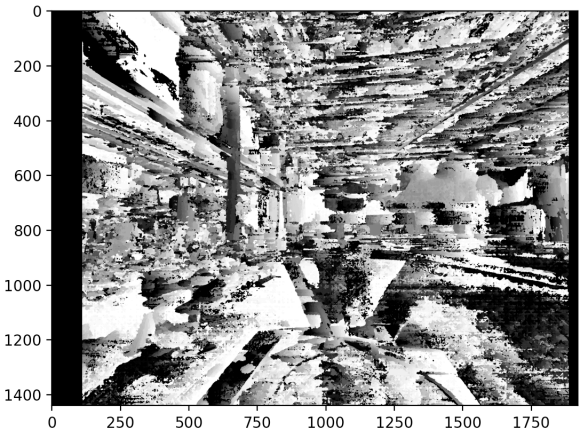


Figure 4.8: Depth map of the shot put environment scene. One can somewhat infer the scene of the shot put circle when studying it up close, but the image is accompanied by a great deal of noise and inconsistencies in depth.

consistency and does rely on a very accurate set of correspondence points, it was found to be the most robust and reliable choice of the three.

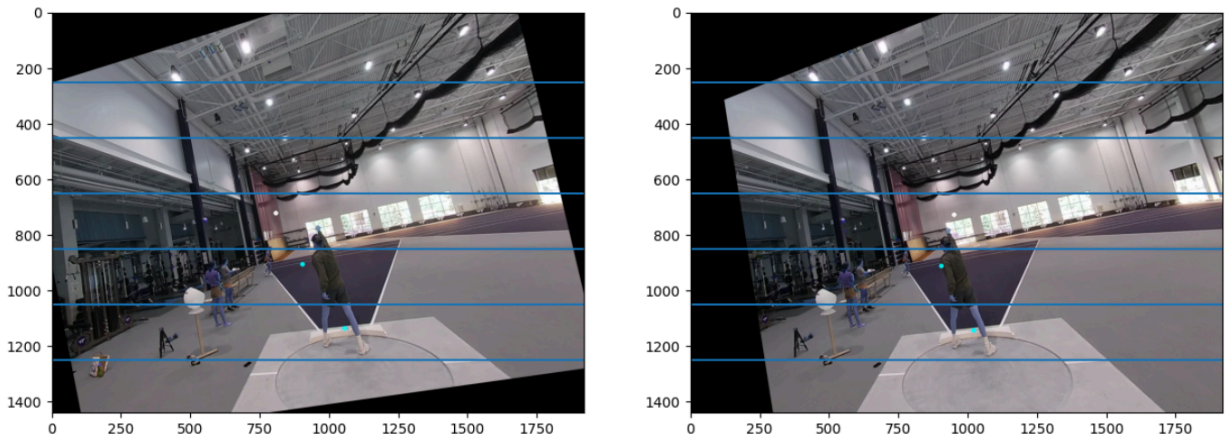
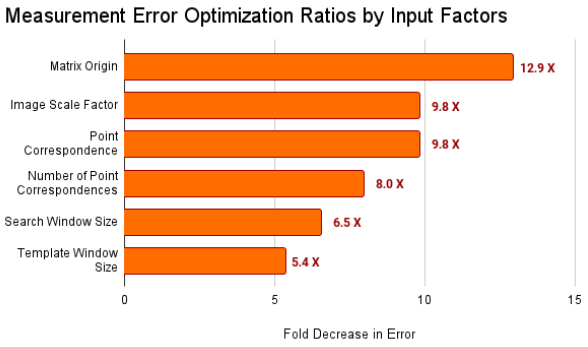


Figure 4.9: Visualization of two rectified images, with lines to demonstrate how the y-axes of the two images are adjusted to correspond to each other. The shot put and stop board points are circled in light blue in each frame. To the human eye, it appears the corresponding coordinates in each image are collinear with respect to the y-axis, but at the pixel level, they are still around 20 pixels different, which affects the triangulation from image to global coordinate, since this assumes a difference of zero pixels between y-values.

After the Hartley-Zisserman method was determined to have the greatest potential for accurate measurement, various input factors within this heuristic were adjusted to study the effects of each, and which combinations resulted in the best results *(See Figure 4.10)*. First, two methods for point correspondences were tested - SIFT for automatic correspondences versus clicking for manual correspondences. While SIFT returned a greater number of point correspondences, it does not perform better than a human manually clicking point correspondences, regardless of the number. Three numbers of manual point correspondences are tested - 9, 16 and 25 points - more points return better results. Refining the point correspondences through template matching leads to slightly better results than not refining as it does provide sub-pixel correspondences, but is not as dramatic as varying other factors. Within the refining technique, the amount the image is upscaled by to find the subpixel accuracies is varied, and scaling up a small amount provides the best results, as scaling up significantly most likely leads to too much reliance on interpolation. Additionally, testing the search window and template window sizes in template matching proved that both a larger template window and larger search window led to better results. Increasing the size of the search and template windows had negligible effects on efficiency.

Once the point correspondences were set, the difference between using the known coordinates to calculate the fundamental matrix first, followed by essential matrix or using the coordinates along with the intrinsic matrices to go straight to the essential matrix are compared. By far, using the findFundamentalMat() function from OpenCV first, and then translating to essential matrix leads to a better error than the alternative direct findEssentialMat() function.



Input Factor	Change in Value/Method	Fold Ratio of Error
Matrix Origin	Essential to Fundamental	12.9
Image Scale Factor	8 to 4 times	9.8
Point Correspondence Method	SIFT to Manual	9.8
Number of Point Correspondences	16 to 25 points	8.0
Search Window Size	20 to 50 pixels	6.5
Template Window Size	4 to 10 pixels	5.4

Figure 4.10: Fold ratios of percent error of measuring shot put distance by adjusting different factors as it relates to the mathematical steps towards three-dimensional reconstruction of the points of interest. The graph demonstrates the magnitudes and effect of each error, whereas the table on the right shows the change performed for each factor. The X represents by how many "times" the average percent error was improved. By far, the best error difference is beginning with the fundamental matrix as opposed to the essential matrix, followed by decreasing the scale factor and using manual point correspondences.

4.2 Accuracy vs. Efficiency

One major tradeoff in any work is comparing the level of correctness with the amount of time to retrieve results. Many steps were taken to improve accuracy or efficiency.

To improve accuracy:

1.) Custom datasets, including additional preprocessing were created to train the deep learning model towards perfecting the detection of one specific object. Datasets included several instances of obscure scenarios of the shot put and augmented data.

2.) A heuristic that combines a detector and predictor, rather than just detector.

3.) The pixel locations during tracking are stored at the sub-pixel level, based on the floating point results from either the detector or predictor.

4.) Accelerations and velocities are recalculated after a string of too many predictions with-

out detections.

5.) The landing point, specifically, is interpolated to find the best prediction for an inter-frame coordinate.

6.) Even though only eight known sets of point correspondences are required for the 8-point algorithm, 25 are required to be clicked by the human to improve the likelihood that at least eight of the matches are of a high accuracy.

7.) After known coordinates are manually clicked, rather than using a match detector such as SIFT, they undergo a refining process to improve matches to a sub-pixel accuracy of an eighth of a pixel.

To improve efficiency:

1.) Several calibrating aspects are performed ahead of time so in real time, less computation is required.

2.) Deep learning models that can identify objects in real time are employed.

3.) All of the mathematical steps for reconstruction, as well as the triangulation of the stop board coordinate are performed ahead of run time, so that only two measurement steps occur during run time.

Ultimately, accuracy is prioritized over efficiency because the program can run in the background. Results should be returned in a rapid manner, yet do not have to be immediate or have to be returned before the next person can throw the shot put. However, this proposed measurement system holds little value if it cannot measure the shot put more accurately than humans, therefore it is worth the extra computation time for precise results.

4.3 Breadth vs. Depth in Object Detection

One discovery through this research is the scarcity of object detection methods with an application-specific focus. Most of the work done in the object detection realm, regardless of the object size, is done in a breadth-first manner, meaning the focus is detecting as many objects as possible from a given frame. But, there is little focus on creating general models that improve the object detection of granular objects or specific subsets of objects.

For example, even the YOLOv5 pre-trained model has a sports ball as one of its 80 classes, the model was only able to detect the shot put at the very beginning of the throw and lost it easily afterwards - performing significantly worse than some non-learning methods. Even when cropping the image to the region of interest and applying superresolution to those pixels, detection of the shot put did not improve.

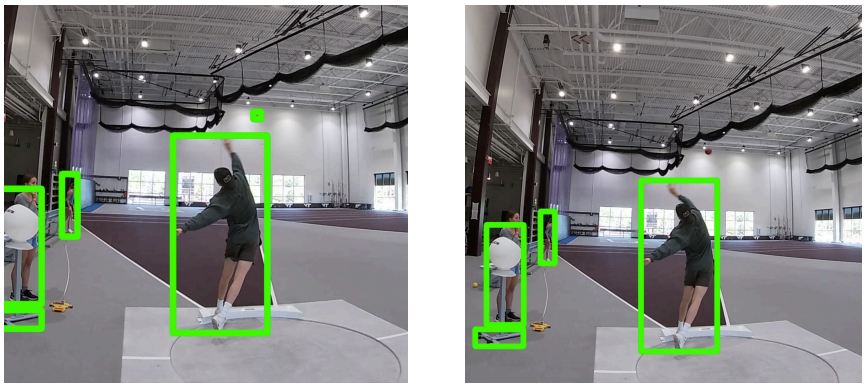


Figure 4.11: Results of sports ball detection on the YOLOv5 pre-trained model. The left frame is the final frame where the shot put is detected; the right frame is an example of the shot put not being detected while it is still a clearly visible object.

When the only focus is on one object, it is not worth the extra computation to segment the image and run detections on every segment, or spend time building lots of pyramids, especially if the location of the object can be predicted. OpenPose is one example [79] of improving human position detection, but there are few robust systems available.

This problem was evident in all levels of testing, whether relating to deep learning techniques or not. For example, the existing object trackers made assumptions that were not necessarily conducive to ball detection, such as relying on the consistency of what the object looks like as the video plays.

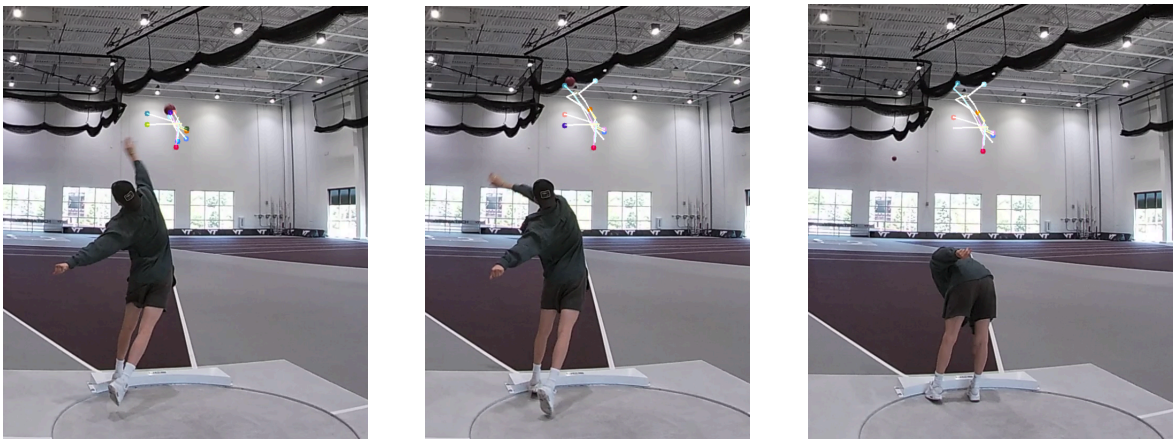


Figure 4.12: Results of applying the Lucas-Kanade tracker towards a shot put throw. In the first image, the tracking is stable, the second image is where the tracking begins to lose its trajectory, and the third image demonstrates how once lost, the tracker is unable to regain the path.

4.4 Presumptive Object Detection

Building on the subject of better application-specific object tracking, there is a need for the ability to only slightly adjust the code when knowing information about the object(s) of interest ahead of runtime. Color-based approaches are common if colors are known, but what about size-based? Location-based? Aspect ratio-based? Deep learning takes care of many of these questions because the network is trained to identify the specific objects, but this requires thousands of labeled images, available CPU size and enough time to train, which may not always be an option.

Yet, there are improvements that can make non-deep learning methods more reliable. For example, the connected components algorithm through OpenCV [92] returns all components found of any shape and size, including the whole image as a component. When testing this algorithm, I later filtered the results based on assumptions I knew about the shot put. However, computation time may be reduced if these filters could be added at the time of calling the function - not guaranteeing a perfect detection of the object of interest, but significantly narrowing down the potential candidates. For example, the figure below shows the difference between all of the components identified, and the result after a few simple filtering mechanisms, which results in a dramatic reduction.

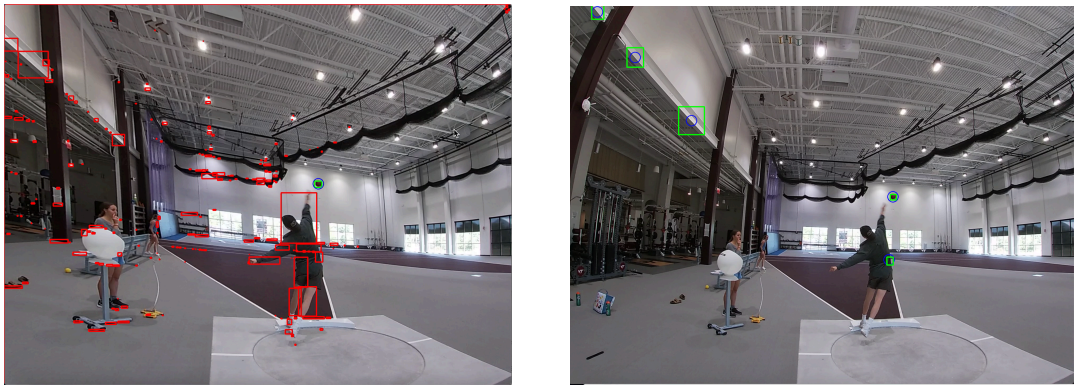


Figure 4.13: A visual comparison of the difference between running the connected components algorithm on a binarized image in a general fashion, and after applying filters on the results of this algorithm. Many irrelevant components are quickly filtered out; however, this could be done during the algorithm, rather than additional step after processing is over.

4.5 Early vs. Late Video Fusion

One deliberate design decision for this system is to run most of the algorithm on the videos in the left and right cameras separately, and only fuse the two at the end once the landing point is found in each camera. This method has several benefits, mainly by overcoming

inconsistencies due to lack of synchronization. When cameras are operating at such a high frame rate, even if they approximately synchronized, the action in corresponding frames may occur at drastically different points. This may not be a problem in cases where there is little movement, but becomes a major issue in applications that involve rapid motion. Because detection and tracking occur separately, the specific frame in which each detection occurs does not matter, as long as each camera returns a valid point of flight initiation and the correct landing point.

However, late fusion also has tradeoffs, especially in edge cases. If the shot put is not detected in a frame, especially at crucial points, the information from the other camera could be used to predict such a point. Early fusion could also be used to provide more robust tracking in cases where the object blends with the background, and may be worth testing in the future. With regard to three-dimensional reconstruction, since the cameras are stationary, early fusion would not require any additional or incremental calibration. Therefore, the three-dimensional coordinate and global path of the shot put could be calculated at each frame with just a minor increase in computation time. The only caveat to early fusion is that the cameras must be highly synchronized for triangulation to be accurate.

4.6 Next Steps

4.6.1 Camera and Frame Rate Analysis

To understand the tradeoffs between a more costly camera and the potential benefits of a higher frame rate, more testing on cameras with both lower and higher frame rates would be valuable. A higher frame rate would likely yield greater accuracy because each approximation occurs in a smaller time step, so less prediction and estimation would be needed. However, it may be computationally too expensive to use a higher frame rate and may not produce a system that works in real time.

A lower frame rate may yield accurate results, while requiring much less time. Testing at a lower frame rate could be performed with the existing videos by adjusting the algorithm to only run the detector every second, third or fourth frames. Additional cameras could be added to determine if accuracy of tracking and measurement is improved with an additional viewpoint. Adding cameras would cost more money and space, but would most likely increase the accuracy of detection in scenarios with occlusion.

4.6.2 Stronger Object Detection Training

There were some limitations to training parameters for YOLOv3 due to the amount of available space of the machine and CPU. The batch size was limited to 8 samples, and the image size was limited to 640x640 pixels. Because my application is a small object detection problem, results would most likely improve if the model were trained with an image size of 1,280 pixels or greater, so that more pixels would be associated with the small object. Additionally, rather than using the YOLOv3 pre-trained model for detection of flight, it may be more efficient to train another model to look for only humans and shot put implements

to avoid added time caused by the identification of trivial objects.

4.6.3 More Consistency in Measurement Results

In order to make this system a viable product in track and field, the measurement results need to provide an accurate measurement more consistently. One suggestion to improve the accuracy of measurement is better and more in-depth camera calibration, where hundreds of videos are taken in numerous different settings to obtain the intrinsic matrices. Exploring a way to perform both intrinsic and extrinsic calibration together in the competition setting would most likely improve results; in this work, intrinsic and extrinsic calibration was performed independently of one another. Additionally, a higher quality camera may result in less noise, which is a main factor of error related to finding the correct fundamental matrix. Even though it may be tedious, because it is only a one time task, it would most likely be beneficial for the user to click more than 25 corresponding points between the images to help lower the error.

4.6.4 Additional Methods to Improve Object Measurement

In the last three years, deep learning has been applied to three-dimensional reconstruction, and while still in the "early stages", this technique holds immense potential. One shortcoming of traditional stereo methods is its struggle to construct "dense and accurate correspondences", and deep learning has the ability to learn much more information in an end-to-end manner [116]. For example, a three-dimensional point cloud could be constructed during calibration using thorough deep learning techniques, leading to more likelihood of an accurate measurement at runtime. Even without deep learning, a more iterative process could be invoked that works ahead of runtime to better refine the fundamental, essential,

and projection matrices.

4.6.5 Edge Cases

While several steps were taken to make this system as robust as possible, further work needs to be done to ensure the system still returns a correct result in worst-case scenarios. For example, in any cases where the point of landing cannot be identified in one or both images due to occlusion from the thrower, there may need to be an alternative solution. Additionally, error checking is needed during scene calibration to make sure the user clicks the right number of points and is able to reset the values if they accidentally click the wrong location. It may also be useful to include suggestions for where to click points for the 8-point algorithm to avoid cases where the points are coplanar or not diverse enough. More video data of throws that travel very small or very large distances would be beneficial for testing purposes, as these types of throws may require adjustments to the algorithm.

4.6.6 Additional Implements and Scenarios

All training and testing was done with shot put videos in indoor competition settings as a starting point, but to increase versatility of the system, training and testing must also include outdoor shot put throws. This may lead to new problems to be solved, such as dealing with natural light fluctuations that may significantly affect the appearance of the shot put or strong winds that could change trajectory. The shot put's bounce after landing may also be quite different in grass or gravel throwing sectors, rather than the hard surface of indoor settings.

Shot put is not the only throwing event in track and field - there are also weight, javelin, hammer and discus events [7]. The throwing implements for each event have different shapes,

some more or less uniform than others. While the measurement process is the same for all, tracking each implement would require different training along with potentially a different path prediction method as they do not all follow an ideal projectile motion. Typically, the javelin, discus and hammer travel much further than the shot put and are affected by aerodynamic forces, leading to a more difficult and even smaller object detection problem, which may require more adjustments.

Average Error of Calculated Shot Put Distance				Use of SIFT	Use of Fundamental Matrix		Refining Point Correspondences			
				no						yes
				no			yes			no
Number of Point Correspondences	Image Scale Factor	Search Window Dimension	Template Window Dimension	both	no	right	both	no	right	no
9	4	20	4	74.32	287.97	287.97	254.57	18.47	19.40	60.07
9	4	20	10	11.10	287.97	287.97	26.55	18.47	13.08	56.65
9	4	20	16	16.94	287.97	287.97	**2.52**	18.47	15.30	62.44
9	4	30	4	752.16	287.97	287.97	76.88	18.47	24.20	85.58
9	4	30	10	60.22	287.97	287.97	27.22	18.47	25.85	135.31
9	4	30	16	**8.33**	287.97	287.97	48.07	18.47	23.03	16.03
9	4	50	4	**4.53**	287.97	287.97	22.78	18.47	**3.15**	47.42
9	4	50	10	**5.05**	287.97	287.97	54.58	18.47	152.60	105.52
9	4	50	16	30.56	287.97	287.97	51.98	18.47	42.66	239.51
9	8	20	4	18.51	287.97	287.97	99.41	18.47	19.33	33.49
9	8	20	10	11.13	287.97	287.97	26.95	18.47	13.04	40.91
9	8	20	16	25.09	287.97	287.97	**2.01**	18.47	14.80	41.44
9	8	30	4	1,283.79	287.97	287.97	77.13	18.47	10.25	38.36
9	8	30	10	308.54	287.97	287.97	44.82	18.47	**3.10**	122.47
9	8	30	16	27.78	287.97	287.97	**8.24**	18.47	22.28	56.57
9	8	50	4	**9.64**	287.97	287.97	**3.31**	18.47	51.57	824.63
9	8	50	10	189.90	287.97	287.97	26.20	18.47	165.61	39.47
9	8	50	16	37.80	287.97	287.97	18.27	18.47	44.34	52.28
9	16	20	4	18.52	287.97	287.97	94.70	18.47	21.03	46.32
9	16	20	10	11.18	287.97	287.97	29.17	18.47	12.95	48.84
9	16	20	16	25.13	287.97	287.97	**3.15**	18.47	14.74	51.07
9	16	30	4	83.94	287.97	287.97	45.01	18.47	13.32	41.21
9	16	30	10	330.18	287.97	287.97	**3.00**	18.47	25.09	50.59
9	16	30	16	47.88	287.97	287.97	**6.51**	18.47	**5.72**	37.35
9	16	50	4	**7.96**	287.97	287.97	**2.68**	18.47	14.56	41.25
9	16	50	10	16.21	287.97	287.97	23.09	18.47	172.47	16.21
9	16	50	16	555.51	287.97	287.97	71.93	18.47	220.16	60.90
16	4	20	4	20.65	110.37	110.37	18.01	39.37	25.93	31.54
16	4	20	10	36.26	110.37	110.37	16.81	39.37	36.15	90.94
16	4	20	16	85.75	110.37	110.37	17.20	39.37	28.40	80.18
16	4	30	4	18.03	110.37	110.37	10.43	39.37	82.98	55.56
16	4	30	10	**7.71**	110.37	110.37	27.69	39.37	81.32	133.78
16	4	30	16	10.63	110.37	110.37	19.40	39.37	14.36	59.66
16	4	50	4	19.21	110.37	110.37	18.93	39.37	**7.42**	12.87
16	4	50	10	116.68	110.37	110.37	15.57	39.37	**4.69**	511.75
16	4	50	16	204.86	110.37	110.37	32.33	39.37	85.82	60.16
16	8	20	4	20.40	110.37	110.37	17.82	39.37	26.95	32.38
16	8	20	10	575.33	110.37	110.37	16.71	39.37	36.26	15.85
16	8	20	16	139.90	110.37	110.37	17.09	39.37	28.04	83.67
16	8	30	4	108.37	110.37	110.37	11.47	39.37	61.37	33.42
16	8	30	10	**5.61**	110.37	110.37	12.74	39.37	74.16	611.53
16	8	30	16	168.19	110.37	110.37	31.72	39.37	18.11	14.63
16	8	50	4	126.87	110.37	110.37	17.74	39.37	59.13	175.16
16	8	50	10	78.68	110.37	110.37	36.01	39.37	58.31	716.91
16	8	50	16	**5.91**	110.37	110.37	58.15	39.37	20.43	288.44
16	16	20	4	16,308.83	110.37	110.37	28.34	39.37	25.82	728.88
16	16	20	10	16.03	110.37	110.37	16.54	39.37	36.54	521.50
16	16	20	16	80.50	110.37	110.37	17.21	39.37	27.82	1,694.24
16	16	30	4	13.88	110.37	110.37	**8.68**	39.37	56.50	912.18
16	16	30	10	**9.03**	110.37	110.37	11.15	39.37	63.74	37.03
16	16	30	16	27.99	110.37	110.37	39.07	39.37	18.55	3,179.85
16	16	50	4	27.84	110.37	110.37	14.40	39.37	**7.48**	52.64
16	16	50	10	**4.34**	110.37	110.37	11.42	39.37	60.89	569.76
16	16	50	16	**2.38**	110.37	110.37	45.33	39.37	**3.35**	78.63
25	4	20	4	24.38	106.89	106.89	12.60	12.48	19.82	39.11
25	4	20	10	10.26	106.89	106.89	10.46	12.48	20.50	54.11
25	4	20	16	33.05	106.89	106.89	10.89	12.48	17.41	48.30
25	4	30	4	383.12	106.89	106.89	12.12	12.48	17.89	46.61
25	4	30	10	32.57	106.89	106.89	11.74	12.48	24.35	40.72
25	4	30	16	79.10	106.89	106.89	22.55	12.48	19.55	53.18
25	4	50	4	31.50	106.89	106.89	**5.96**	12.48	34.43	25.96
25	4	50	10	17.80	106.89	106.89	**7.01**	12.48	18.91	177.90
25	4	50	16	23.04	106.89	106.89	17.88	12.48	34.85	80.25
25	8	20	4	69.47	106.89	106.89	12.78	12.48	20.16	95.94
25	8	20	10	**9.91**	106.89	106.89	10.47	12.48	20.19	31.56
25	8	20	16	16.92	106.89	106.89	10.97	12.48	17.26	39.90
25	8	30	4	71.70	106.89	106.89	18.06	12.48	14.32	96.37
25	8	30	10	376.96	106.89	106.89	14.22	12.48	25.51	146.64
25	8	30	16	**7.10**	106.89	106.89	13.43	12.48	30.93	50.19
25	8	50	4	301.98	106.89	106.89	103.13	12.48	15.21	118.29
25	8	50	10	18.01	106.89	106.89	**8.68**	12.48	23.07	128.08
25	8	50	16	14.53	106.89	106.89	711.85	12.48	27.13	34.93
25	16	20	4	49.78	106.89	106.89	12.85	12.48	16.17	16.67
25	16	20	10	15.95	106.89	106.89	10.21	12.48	20.14	52.30
25	16	20	16	18.12	106.89	106.89	10.54	12.48	17.16	928.35
25	16	30	4	**5.04**	106.89	106.89	48.07	12.48	14.35	20.53
25	16	30	10	**6.74**	106.89	106.89	27.67	12.48	32.18	40.53
25	16	30	16	**9.39**	106.89	106.89	22.75	12.48	27.54	45.42
25	16	50	4	48.02	106.89	106.89	793.64	12.48	15.41	40.45
25	16	50	10	**3.91**	106.89	106.89	36.88	12.48	23.10	32.66
25	16	50	16	13.42	106.89	106.89	16.32	12.48	23.00	12.82

Figure 4.14: Pivot table to visualize the average error out of 1.0 obtained from the numerous combinations towards the reconstruction and measurement process. Highlighted are any error values less than 10.0. The lowest error is 2.01 and in general, scenarios where both the left and right coordinates are refined performs better, as well as using a larger number of point correspondences. All calculations in this table are done without removing any possible outliers.

Chapter 5

Conclusions

For the shot put throw to be accurately tracked and the distance to be accurately measured, a robust tracking system combining deep learning and kinematic interpolations along with camera properties and stereo vision mathematics was created. Many existing works in object tracking through both traditional and deep learning methods rely on the visual consistency of objects across frames and are more focused on an average detection of several objects, rather than almost perfect detection of an object of interest known ahead of time. Additionally, object detection and tracking models lack quality results as they pertain to less than ideal objects, including those that are occluded and small. By training custom datasets with augmented data, performance of object detection dramatically improved. Tracking is further enhanced by the ability to predict the next point, so there is no reliance on perfect identification of the object of interest in every frame. These enhancements also allow one to narrow down the objects of interest in tracking, so computation time is not wasted by a model identifying trivial objects. Allowing the user to manually enter points ahead of time increases the likelihood of accuracy in calibrating the cameras and performing correct reconstruction into three dimensions.

This system I created can track and measure the shot put in an average of 4.5 minutes per throw, with the only manual work being the calibration before any throwers step into the throwing circle. The system includes algorithms that enhance functionality so small objects, less than ideal objects, and occluded objects can be tracked and the distance traveled can

be measured. It can calculate a measurement within 82% accuracy and suggestions for lowering accuracy include more thorough intrinsic calibration and a more iterative or deep learning extrinsic calibration process. Between the two GoPro cameras and two tripods, the equipment of the system costs around $500. Not only is this work useful in the sport of track and field, but it is also useful in finding shortcomings among machine learning and computer vision as it pertains to real-world problems. These findings can be applied to other examples of tracking or measuring small, less than ideal objects, so it could be used for other sports and possibly in the fields of medicine or autonomous vehicles.

Bibliography

[1] Clare Murphy Smith. *An empirical investigation of the patronage behavior of nutrition oriented consumers*. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1988.

[2] Sport. in Cambridge Dictionary, *Cambridge University Press*, 2021. [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/sport, Accessed on: Mar 25, 2021.

[3] Brad R. Humphreys and Jane E. Ruseski. Estimates of the Size of the Sports Industry in the United States, 2008.

[4] Gerald A. Carr. *Fundamentals of track and field*. Human Kinetics, 2nd ed edition.

[5] NCAA. Estimated probability of competing in college athletics, April 2020. Available: http://www.ncaa.org/about/resources/research/estimated-probability-competing-college-athletics, Accessed Jun 16, 2021.

[6] USA Track and Field. About USATF, 2021. Available: https://usatf.org/about, Accessed Jun 16, 2021.

[7] World Athletics. C2.1 Technical Rules, December 2021. Available: https://www.worldathletics.org/about-iaaf/documents/book-of-rules, Accessed Dec 23, 2021.

[8] Donatella Di Corrado, Elena Pellarin, and Tiziano Alessandro Agostini. The phenomenon of social influence on the football pitch: Social pressure from the crowd on referees' decisions. *Review of Psychology*, 18(1):33–36, 2011.

[9] Tom Fleischman. Track deaths rare, but happen, December 2014. Available: https://www.ithacajournal.com/story/news/local/2014/12/10/cornell-track-field-death/20218737/, Accessed July 15, 2021.

[10] BBC News. Javelin accident kills German athletics official. *BBC News*, August 2012.

[11] JR Landry. How is the shot put throw measured? Available: https://www.sportsrec.com/6587609/how-is-the-shot-put-throw-measured, Accessed Dec 29, 2021.

[12] Mark Heckel. Measuring at the circle, 2014. Available: https://www.youtube.com/watch?v=kpioqEOWbNY, Accessed on: Dec 29, 2021.

[13] Iring Koch, Edita Poljac, Hermann Müller, and Andrea Kiesel. Cognitive structure, flexibility, and plasticity in human multitasking—An integrative review of dual-task and task-switching research. *Psychological Bulletin*, 144(6):557–583, June 2018.

[14] Amer Al-Rahayfeh and Miad Faezipour. Enhanced frame rate for real-time eye tracking using circular hough transform. In *2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–6, 2013.

[15] Yan Baodong. Hawkeye technology using tennis match. *Computer Modelling and New Technologies*, 18:400–402, November 2014.

[16] Thomas Dohmen and Jan Sauermann. REFEREE BIAS: REFEREE BIAS. *Journal of Economic Surveys*, 30(4):679–695, September 2016.

[17] Cedric Gottschalk, Stefan Tewes, and Benjamin Niestroj. The Innovation of Refereeing in Football Through AI. *International Journal of Innovation and Economic Development*, 6(2):35–54, 2020.

[18] World Athletics. Senior Outdoor 2021 Shot Put Women Top List, December 2021. Available: https://www.worldathletics.org/records/toplists/throws/shot-put/outdoor/women/senior/2021, Accessed Dec 29, 2021.

[19] Chad M. Oldfather and Matthew M. Fernholz. Comparative Procedure on a Sunday Afternoon: Instant Replay in the NFL as a Process of Appellate Review. *Indiana Law Review*, 43(1):45–78, 2009.

[20] Emily E Cust, Alice J Sweeting, Kevin Ball, and Sam Robertson. Machine and deep learning for sport-specific movement recognition: a systematic review of model development and performance. *Journal of Sports Sciences*, 37(5):568–600, March 2019.

[21] Allison L. Clouthier, Gwyneth B. Ross, and Ryan B. Graham. Sensor Data Required for Automatic Recognition of Athletic Tasks Using Deep Neural Networks. *Frontiers in Bioengineering and Biotechnology*, 7:473, January 2020.

[22] Yusuke Mizushina, Wataru Fujiwara, Tomoaki Sudou, Charith Lasantha Fernando, Kouta Minamizawa, and Susumu Tachi. Interactive instant replay: sharing sports experience using 360-degrees spherical images and haptic sensation based on the coupled body motion. In *Proceedings of the 6th Augmented Human International Conference*, pages 227–228, Singapore Singapore, March 2015. ACM.

[23] R. Glenn Cummins and Dustin Hahn. Re-presenting Sport: How Instant Replay and Perceived Violence Impact Enjoyment of Mediated Sports. *Mass Communication and Society*, 16(6):787–807, November 2013.

[24] S. Christopher Szczerban. Tackling Instant Replay: A Proposal to Protect the Competitive Judgments of Sports Officials. *Virginia Sports and Entertainment Law Journal*, 6(2):277–332, 2006.

[25] Craig Calcaterra. Apparently, instant replay is really expensive. *NBC Sports*, Jun 2012.

[26] Craig Berman. Pros & cons of instant replay in sports, Jul 2011. Available: https://www.sportsrec.com/pros-cons-instant-replay-sports-8681745.html, Accessed Dec 29, 2021.

[27] Otto Kolbinger and Melanie Knopp. Video kills the sentiment—Exploring fans' reception of the video assistant referee in the English premier league using Twitter data. *PLOS ONE*, 15(12):e0242728, December 2020.

[28] Dan Russell. Laws of the Game 20/21. *The International Football Association Board*, June 2020.

[29] Associated Press. Video assistant referee decisions at World Cup to have written explanations. *ESPN*, April 2018.

[30] Jake Michaels. 'There are just no mistakes happening': Hawk-Eye Live gains more support at Australian Open. *ESPN*, February 2021.

[31] Basilio Pueo, Jose J. Lopez, Jose M. Mossi, Adrian Colomer, and Jose M. Jimenez-Olmedo. Video-based system for automatic measurement of barbell velocity in back squat. *Sensors*, 21(3), 2021.

[32] Adrian Rosebrock. Ball Tracking with OpenCV. *PyImageSearch*, September 2015. Available: https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/, Accessed Dec 31, 2021.

[33] Murk J. Bottema and John P. Slavotinek. Detection and classification of lobular and dcis (small cell) microcalcifications in digital mammograms. *Pattern Recognition Letters*, 21(13-14):1209–1214, December 2000.

[34] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks*, pages 2809–2813, 2011.

[35] Jonathan M. Peake, Graham Kerr, and John P. Sullivan. A Critical Review of Consumer Wearables, Mobile Applications, and Equipment for Providing Biofeedback, Monitoring Stress, and Sleep in Physically Active Populations. *Frontiers in Physiology*, 9:743, June 2018.

[36] Wearable technology. in *Dictionary.com*, Apr. 2021. [Online]. Available: https://www.dictionary.com/browse/wearable-technology, Accessed on: Apr. 26, 2021.

[37] Juri Taborri, Eduardo Palermo, and Stefano Rossi. Automatic Detection of Faults in Race Walking: A Comparative Analysis of Machine-Learning Algorithms Fed with Inertial Sensor Data. *Sensors (Basel, Switzerland)*, 19(6), March 2019.

[38] E. H. Chi. Introducing wearable force sensors in martial arts. *IEEE Pervasive Computing*, 4(3):47–53, July 2005. Conference Name: IEEE Pervasive Computing.

[39] Tony Luczak, Reuben Burch, Edwin Lewis, Harish Chander, and John Ball. State-of-the-art review of athletic wearable technology: What 113 strength and conditioning coaches and athletic trainers from the USA said about technology in sports. *International Journal of Sports Science & Coaching*, 15(1):26–40, February 2020.

[40] Jonas Lutz, Daniel Memmert, Dominik Raabe, Rolf Dornberger, and Lars Donath. Wearables for Integrative Performance and Tactic Analyses: Opportunities, Challenges, and Future Directions. *International Journal of Environmental Research and Public Health*, 17(1):59, December 2019.

[41] Y Ciu. Intelligent wireless monitoring system for foul play in a walking race based on

UWB. In *Proceedings of the 2nd ETP/IITA Conference on Telecommunication and Information*, Phuket, Thailand, April 2011.

[42] Sportvu. *Stats Perform*, Available: https://www.statsperform.com/team-performance/football-performance/optical-tracking/.

[43] Ben Dornan. FINA APPROVES USE OF UNDERWATER VIDEO TECHNOLOGY AT OLYMPICS AND WORLD CHAMPS, March 2020.

[44] History & Evolution of Sports Broadcasting | Be On Air, January 2015.

[45] R. Saravanan and Pothula Sujatha. A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 945–949, Madurai, India, June 2018. IEEE.

[46] A. Mcintyre, Joel Brooks, J. Guttag, and J. Wiens. Recognizing and Analyzing Ball Screen Defense in the NBA. In *In Proceedings of the MIT Sloan Sports Analytics Conference*, volume 1001, page 48104, Boston, MA, March 2016.

[47] Rajiv Maheswaran. Transcript of "The math behind basketball's wildest moves". *TED*, Apr 2021.

[48] Courtvision. *Los Angeles Clippers* powered by AWS, Available: https://www.clipperscourtvision.com/.

[49] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, November 2006.

[50] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27(7):773–780, may 2006.

[51] Changjia Tian, Varuna De Silva, Michael Caine, and Steve Swanson. Use of Machine Learning to Automate the Identification of Basketball Strategies Using Whole Team Player Tracking Data. *Applied Sciences*, 10(1):24, December 2019.

[52] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[54] Daniel Berrar. Cross-Validation. In *Encyclopedia of Bioinformatics and Computational Biology*, pages 542–545. Elsevier, 2019.

[55] Juan Du. Understanding of Object Detection Based on CNN Family and YOLO. *Journal of Physics: Conference Series*, 1004:012029, April 2018.

[56] Shantanu Gangal and Sangram Raje. The hawkeye technology. *Computer Science and Engineering (CSE) Department, Indian Institute of Technology, Bombay*, 2007.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[58] J. Redmon. Darknet: Open Source Neural Networks in C. [Online]. Available: http://pjreddie.com/darknet.

[59] Young Yoon, Heesu Hwang, Yongjun Choi, Minbeom Joo, Hyeyoon Oh, Insun Park, Keon-Hee Lee, and Jin-Ha Hwang. Analyzing Basketball Movements and Pass Relationships Using Realtime Object Tracking Techniques Based on Deep Learning. *IEEE Access*, 7:56564–56576, 2019.

[60] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[61] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. cite arxiv:1804.02767Comment: Tech Report.

[62] Jing Tao, Hongbo Wang, Xinyu Zhang, Xiaoyu Li, and Huawei Yang. An object detection system based on YOLO in traffic scene. In *2017 6th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 315–319, Dalian, China, October 2017. IEEE.

[63] Jihyun Seo, Jaewon Sa, Younchang Choi, Yongwha Chung, Daihee Park, and Hakjae Kim. A YOLO-based Separation of Touching-Pigs for Smart Pig Farm Applications. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pages 395–401, February 2019. ISSN: 1738-9445.

[64] Zhiguang Cao, Tingbo Liao, Wen Song, Zhenghua Chen, and Chongshou Li. Detecting the shuttlecock for a badminton robot: A YOLO based approach. *Expert Systems with Applications*, 164:113833, February 2021.

[65] Stephanie Kovalchik. AO Leaderboard - Forehand Speeds. *Stats On the T*, November 2016. Available: http://on-the-t.com/2016/11/26/aoleaderboard-forehand-speed/.

[66] Chris Chase. In case you were wondering how fast a football can be thrown by a human being. *USA TODAY Sports*, March 2014.

[67] Andrew Simon. The 11 hardest-throwing rotations for 2019. *MLB Advanced Media, LP*, January 2019.

[68] John C. Abbott, John P. Wagle, Kimitake Sato, Keith Painter, Thaddeus J. Light, and Michael H. Stone. Validation of Inertial Sensor to Measure Barbell Kinematics across a Spectrum of Loading Conditions. *Sports*, 8(7):93, June 2020.

[69] Georg Waltner, Thomas Mauthner, and Horst Bischof. Improved sport activity recognition using spatio-temporal context. In *Proc. DVS-Conference on Computer Science in Sport (DVS/GSSS)*. ., 2014.

[70] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. In *9th International Conference on Advances in Computing and Information Technology (ACITY 2019)*, pages 119–133. Aircc Publishing Corporation, December 2019.

[71] Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny object detection in aerial images. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3791–3798, 2021.

[72] Lisha Cui. MDSSD: multi-scale deconvolutional single shot detector for small objects. *CoRR*, abs/1805.07009, 2018.

[73] Chenhongyi Yang, Zehao Huang, and Naiyan Wang. Querydet: Cascaded sparse query for accelerating high-resolution small object detection. *CoRR*, abs/2103.09136, 2021.

[74] Fan Zhang, Bo Du, Liangpei Zhang, and Miaozhong Xu. Weakly supervised learning based on coupled convolutional neural networks for aircraft detection. *IEEE Transactions on Geoscience and Remote Sensing*, 54(9):5553–5563, 2016.

[75] Daniel Ho, Eric Liang, and Richard Liaw. 1000x Faster Data Augmentation. *Berkeley Artificial Intelligence Research*, June 2019.

[76] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2):125, Feb 2020.

[77] Sebastien C. Wong, Adam Gatt, Victor Stamatescu, and Mark D. McDonnell. Understanding data augmentation for classification: when to warp? *CoRR*, abs/1609.08764, 2016.

[78] Martin U. Schlegel and Craig Hill. The Reach of Sports Technologies. In Sascha L. Schmidt, editor, *21st Century Sports: How Technologies Will Change Sports in the Digital Age*, pages 91–110. Springer International Publishing, Cham, 2020.

[79] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *arXiv:1812.08008 [cs]*, May 2019. arXiv: 1812.08008.

[80] P. K. Santhosh and B. Kaarthick. An Automated Player Detection and Tracking in Basketball Game. *Computers, Materials & Continua*, 58(3):625–639, 2019.

[81] M. Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, pages 3099–3104, The Hague, Netherlands, 2004. IEEE.

[82] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, Baden-Baden, Germany, June 2011. IEEE.

[83] F. Bruno, G. Bianco, M. Muzzupappa, S. Barone, and A.V. Razionale. Experimentation of structured light and stereo vision for underwater 3D reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(4):508–518, July 2011.

[84] Seunghun Jin, Junguk Cho, Xuan Dai Pham, Kyoung Mu Lee, Sung-Kee Park, Munsang Kim, and Jae Wook Jeon. FPGA Design and Implementation of a Real-Time Stereo Vision System. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(1):15–26, January 2010.

[85] Stephen Maybank. *Theory of reconstruction from image motion.* Number 28 in Springer series in information sciences. Springer-Verlag, Berlin ; New York, 1993.

[86] Raden Arief Setyawan, Rudy Sunoko, Mochammad Agus Choiron, and Panca Mudji Rahardjo. Implementation of Stereo Vision Semi-Global Block Matching Methods for Distance Measurement. *Indonesian Journal of Electrical Engineering and Computer Science*, 12(2):585, November 2018.

[87] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[88] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[89] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[90] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[91] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011.

[92] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[93] Aaron Bobick. Cs 4495 computer vision n-views (2) – essential and fundamental matrices. https://faculty.cc.gatech.edu/ afb/classes/CS4495-Fall2013/slides/CS4495-09-TwoViews-2.pdf, 2013. *School of Interactive Computing, Georgia Tech*, Accessed on: 2022-03-22.

[94] Lynn Abbott. Depth estimation, 2020. *Virginia Tech*, Accessed on: 2022-03-26.

[95] M. Domnguez-Morales, A. Jimnez-Fernndez, R. Paz-Vicente, A. Linares-Barranco, and G. Jimnez-Moreno. Stereo Matching: From the Basis to Neuromorphic Engineering. In Asim Bhatti, editor, *Current Advancements in Stereo Vision*. InTech, July 2012.

[96] Teresa Cristina de Sousa Azevedo. *3D Object Reconstruction using Computer Vision: Reconstruction and Characterization Applications for External Human Anatomical Structures*. Dissertation, Universidade do Porto, Porto, Portugal, 2012.

[97] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine vision and applications*, 12(1):16–22, 2000.

[98] Guido Gerig. Image rectification (stereo). http://www.sci.utah.edu/g̃erig/CS6320-S2012/Materials/CS6320-CV-F2012-Rectification.pdf, 2012. *University of Utah*, Accessed on: 2022-03-21.

[99] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[100] Anthony P. Badali, Yahui Zhang, Peter Carr, Paul J. Thomas, and Richard I. Hornsey. Scale factor in digital cameras. page 59692B, Toronto, Canada, September 2005.

[101] M. Bertozzi and A. Broggi. GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1):62–81, January 1998.

[102] Christoph Bert, Katherine G. Metheany, Karen Doppke, and George T. Y. Chen. A phantom evaluation of a stereo-vision surface imaging system for radiotherapy patient setup: Evaluation of a stereo-imaging patient setup system. *Medical Physics*, 32(9):2753–2762, August 2005.

[103] Odysseas Sekkas, Vassileios Tsetsos, Aggelos Biboudis, Evangelos Zervas, Nikolaos Silvestros, Stathes Hadjiefthymiades, and Angelos Batistakis. MobiXeyes: Real-time Stereo Vision Technology for Racket Sports. *Procedia Engineering*, 112:546–551, 2015.

[104] Haizhen Li and Baojun Zhang. Application of integrated binocular stereo vision measurement and wireless sensor system in athlete displacement test. *Alexandria Engineering Journal*, 60(5):4325–4335, October 2021.

[105] GoPro Action Cameras: HD Video Cameras for Sports + Adventure. *GoPro Inc.*, Available: https://gopro.com/en/us/shop/cameras.

[106] Satya Mallick. Object Tracking using OpenCV (C++/Python). *Learn OpenCV*, February 2017.

[107] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, page 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[108] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[109] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31 Vol.2, 2004.

[110] Federico Bolelli, Stefano Allegretti, Lorenzo Baraldi, and Costantino Grana. Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling. *IEEE Transactions on Image Processing*, 29:1999–2012, 2020.

[111] Glenn Jocher, Yonghye Kwon, guigarfr, perry0418, Josh Veitch-Michaelis, Ttayu, Daniel Suess, Fatih Baltacı, Gabriel Bianconi, IlyaOvodov, Marc, e96031413, Chang Lee, Dustin Kendall, Falak, Francisco Reveriano, FuLin, GoogleWiki, Jason Nataprawira, Jeremy Hu, LinCoce, LukeAI, NanoCode012, NirZarrabi, Oulbacha Reda, Peretz Cohen, Piotr Skalski, SergioSanchezMontesUAM, Shiwei Song, and Timothy M. Shead. ultralytics/yolov3: v9.6.0 - YOLOv5 v6.0 release compatibility update for YOLOv3, November 2021.

[112] Labelbox. [Online]. Available: https://labelbox.com, Accessed on: Dec. 5, 2021.

[113] Integral approximations. Available: https://www.mathsisfun.com/calculus/integral-approximations.html, Accessed on: Dec 30, 2021.

[114] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981.

[115] Dave Greenwood. triangulation.py. https://gist.github.com/davegreenwood/, 2018.

[116] Anny Yuniarti and Nanik Suciati. A Review of Deep Learning Techniques for 3D Reconstruction of 2D Images. In *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, pages 327–331, Surabaya, Indonesia, July 2019. IEEE.

Appendices

Appendix A

Data Collection

Description of Video	# of Tripods	Tripod Height (cm)	Baseline (cm)	Camera/Tripod Position	Ball Color	Lights	Measurement of Throw (cm)
Shot put in center	1	190.5	24	Behind	Yellow	On	779
Shot put out of bounds right	1	190.5	24	Behind	Yellow	On	N/A
Shot put left of center	1	190.5	24	Behind	Yellow	On	1,104
Shot put right of center	1	190.5	24	Behind	Yellow	On	1,108
Shot put very close	1	190.5	24	Behind	Yellow	On	305
Throwing shot put close	1	190.5	24	Behind	Maroon	On	437
Throwing shot put far	1	190.5	24	Behind	Maroon	On	821
Shot put in center	2	190.5	292	Behind	Yellow	On	647
Shot put out of bounds right	2	190.5	292	Behind	Yellow	On	N/A
Shot put left of center	2	190.5	292	Behind	Yellow	On	1,250
Shot put right of center	2	190.5	292	Behind	Yellow	On	1,210
Throwing shot put close	2	190.5	292	Behind	Maroon	On	508
Throwing shot put far	2	190.5	292	Behind	Maroon	On	578
Throwing shot put close	2	160.0	292	Behind	Maroon	On	636
Throwing shot put far	2	160.0	292	Behind	Maroon	On	1,002
Throwing shot put out of bounds	2	160.0	292	Behind	Maroon	On	N/A
Shot put in center	2	190.5	457	On Line	Yellow	On	731
Shot put out of bounds right	2	190.5	457	On Line	Yellow	On	N/A
Shot put right of center	2	190.5	457	On Line	Yellow	On	1,126
Throwing shot put close	2	190.5	457	On Line	Maroon	On	573
Throwing shot put far	2	190.5	457	On Line	Maroon	On	1,003
Shot put in center	2	190.5	457	On Line	Yellow	Dim	723
Throwing shot put far	2	190.5	457	On Line	Maroon	Dim	707
Throwing shot put close	2	190.5	414	Behind	Maroon	On	407
Human stepping out of bounds	2	190.5	414	Behind	Maroon	On	N/A
Human stepping out of bounds	2	190.5	414	Behind	Maroon	On	N/A
Human stepping out of bounds	2	190.5	414	Behind	Maroon	On	N/A

Camera/Tripod Position = "Behind" indicates they are located behind the thrower.

Camera/Tripod Position = "On line" indicates they are located on the line of measurement.

Dim = Facility lighting is dimmed.

Figure A.1: Table of information from each recorded video - 27 rows are shown because each row has a corresponding left camera and right camera video, resulting in 54 total videos. Subjects of the videos are Brian Baker, Clint Gault, Sarah Edwards, Sara Freix, and myself - Ashley Smith.

Shot Put Tracking Dictionary

Key: Frame Number

Value:

{

predicted x_position: $\text{x\_position}_{\text{pf}} + \text{x\_velocity}_{\text{pf}} + \frac{1}{2}\,\text{x\_acceleration}_{\text{pf}}$

predicted y_position: $\text{y\_position}_{\text{pf}} + \text{y\_velocity}_{\text{pf}} + \frac{1}{2}\,\text{y\_acceleration}_{\text{pf}}$

detected x_position: best YOLO detected x_point

detected y_position: best YOLO detected y_point

x_position:

if best YOLO detected point matches search criteria: detected x_position

else: predicted x_position

y_position:

if best YOLO point matches search criteria: detected y_position

else: predicted y_position

x_velocity: $\text{x\_position} - \text{x\_position}_{\text{pf}}$

y_velocity: $\text{y\_position} - \text{y\_position}_{\text{pf}}$

velocity: $\sqrt{(x\_velocity)^2 + (y\_velocity)^2}$

x_acceleration: $\text{x\_velocity} - \text{x\_velocity}_{\text{pf}}$

y_acceleration: $\text{y\_velocity} - \text{y\_velocity}_{\text{pf}}$

acceleration: $\sqrt{(x\_acceleration)^2 + (y\_acceleration)^2}$

angle:

if x_velocity != 0: $tan^{-1}\left(\frac{y\_velocity}{x\_velocity}\right)$ **else:** Null

is_predicted:

if (x_position and y_position) are predicted: True **else:** False

}

Legend: pf = Previous Frame = Frame Number - 1

Figure A.2: An outline of the nested dictionary structure built to keep track of shot put detections, predicted locations, and important metrics contained in each camera frame.