WORCESTER POLYTECHNIC INSTITUTE

MAJOR QUALIFYING PROJECT

The Detection and Parametric Estimation of Fast-Moving Objects for Smart Home Plate Application

Authors:

Theodore MacLeod

Dominic Palermo

Michael Panicci

Advisor:

Prof. Donald Brown

A paper submitted in fulfillment of the Major Qualifying Project

April 25, 2019

Table of Contents

Table of Figures

Table of Tables

Abstract

This project developed and tested a proof-of-concept of a smartphone-based system for classification of balls and strikes in the game of baseball. To demonstrate the feasibility of this idea, an automated system was developed to determine if a pitch in the game of baseball was a ball or strike. The system consisted of six main steps: data collection, data pre-processing, image classification, object detection, parametric estimation, and pitch classification. Data collection consisted of recording pitches at 240 frames per second at 1080p resolution with an iPhone 8+ camera facing upwards from a home plate. In the data pre-processing stage, the frames were extracted from each video and then sorted into two categories: baseball present and baseball not present. Frames were classified with a convolutional neural network trained to automatically classify whether a baseball was present or baseball was not present. For object detection, the baseball's pixel width and (x, y) pixel location were determined by removing noise and sending the frames through a sequence of filtering methods. For parametric estimation, equations were empirically derived to estimate the 3D (x, y, z) coordinates of the baseball in each frame in units of inches from the previously estimated (x, y) pixel location and pixel width. These estimates were then extrapolated from multiple frames to then calculate a best fit line for the path of the baseball for a given pitch. Finally, the pitch was classified as a ball or a strike by determining if the best fit line intersected the strike zone volume. If an intersection was detected, the pitch was classified as a strike. Otherwise, it was classified as a ball. The system successfully outputs a pitch classification utilizing the input from a smartphone camera.

Executive Summary

In the game of baseball, current professional systems used to call balls and strikes are expensive and typically require careful and professional installation. These systems are not practical for applications outside of the professional leagues, so an umpire's call is never able to be examined for validity. This opens a gap for an inexpensive and accurate solution to calling balls and strikes. The solution was to develop an inexpensive and accurate way to detect and parametrically estimate the baseball's location using video captured by a smartphone installed in a home plate. The chosen approach was to examine currently implemented systems, which consisted of inertial measurement units, camera-based, and radar-based systems. After investigating the prior art, other potential system implementations were investigated, such as radio-frequency identification tags, hall effect sensors, infrared sensors, ultrasonic sensors, and smartphone cameras utilizing machine learning techniques. A system based on a smartphone camera and machine learning techniques proved to be the ideal solution after completing a value analysis that compared each option.

An iPhone 8+ with a 240 frames per second HD video recording capability and fisheye lens was placed in a custom home plate and was used to record pitches. Six hundred 1920x1080 pixel frames were extracted in greyscale, downscaled to 200x200 pixels, and sorted into two categories: baseball present and baseball not present. A training dataset of 480 frames was used to train a convolutional neural network (CNN) and a testing dataset of 120 frames was used to test the CNN's accuracy. A testing accuracy of approximately 90% was achieved. Once a baseball is detected by the trained CNN, the next step processes full-resolution grayscale frames to estimate the baseball's (x, y) pixel coordinates as well as the pixel width of the baseball. This process was automated and coded in Python. The frames classified as containing a baseball had the background removed and were filtered to remove noise. The remaining pixel clusters, one of which being the baseball, were sent through an area and curvature constraint outputting the baseball's width and (x, y) center location in pixels. Through controlled experiments with known ground-truth baseball positions, equations were empirically generated to derive the relationship between the baseball's width and (x, y) pixel location in the image to the baseball's real world (x, y, z) coordinates in inches. From here, a 3D best fit line was drawn through the path of the baseball by using Principal Component Analysis. Lastly, if the baseball path intersected the

strike zone volume, the pitch was classified as a strike, and if not, the pitch was classified as a ball. By going from raw smartphone video to an outputted binary pitch classification of ball or strike, this project provided a proof-of-concept study establishing the framework for an inexpensive and accurate system for calling balls and strikes in the game of baseball.

1. Introduction

The Major League Baseball Association (MLB) uses four umpires to manage a professional baseball game. The umpire behind the home plate judges if a pitch is a strike or a ball. A strike is a pitch that intersects the strike zone volume seen below in Figure 1, and a ball is a pitch that does not. The validity of an umpire's call can currently be examined with professional systems. These professional systems are expensive and require careful, professional, and often permanent setup. These systems are not practical for lower-level baseball leagues with less money or less staff. Therefore, this project fills a gap in the market by establishing an accurate and inexpensive system that can call balls and strikes.

Figure 1: *Volume of strike, according to batters' dimensions [1].*

The strike zone volume varies depending on the player, but its cross-sectional surface area is always fixed. The dimensions used for the cross-sectional surface area come from the dimensions of the home plate shown in Figure 2. Defining this area is crucial to ultimately determining a strike or ball. Figure 1 describes how the volume of the strike zone would change depending on player height, by showing the specific strike zone for the player in that image. The strike zone upper height lines up with the midpoint of the batter's chest, and the lower height

lines up with the bottom of the batter's kneecap. That height difference is what defines the strike zone volume, but this dimension is approximated by the umpire during gameplay, which adds in a degree of subjectivity.

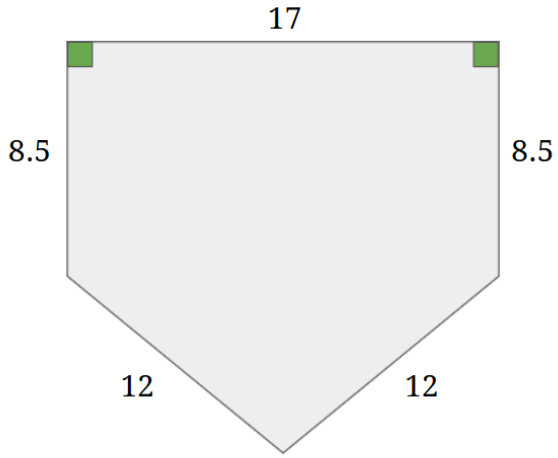

Figure 2: *Dimensions of an MLB regulation home plate (all measurements in inches) [2]*.

1.1 Project Statement and Justification

While a full study of the prior art is provided in Section 2, the following table summarizes the various systems available at the time of this report that can be used for automatically determining balls and strikes in baseball.

Table 1: *Summary of prior art seen in Section 2*

Product	Estimated Price	Setup	Used In-Game
PITCHf/x	$30,000+	Professional	yes
Strike Smart Baseball	$129.99	Amateur	no
PitchTracker	$99+	Amateur	no
TrackMan	$30,000+	Professional	yes
FlightScope Strike	$18,000	Professional	yes
Rapsodo Baseball	$3,000	Professional	yes

From Table 1 above, there was no inexpensive and easy-to-use system that could remove the ambiguity of pitch calls in-game. Therefore, the purpose of this project was to develop an inexpensive and accurate way to detect and parametrically estimate a baseball's location. Only MLB teams have the means and resources to use high end camera systems during practice and play because these systems are expensive and typically unavailable to non-professional teams. Currently, there is no inexpensive and easy-to-use system which removes ambiguity of pitch calls. When non-professional baseball teams practice, hiring an umpire to call balls and strikes is not feasible, so teams lack the precision of a professional. The precision of these professionals is still subject to human error and influenced by the professional players in the game. "While the strike zone is clearly set by the MLB rulebook, good pitchers and catchers will work together to flirt with the edges of an umpire's strike zone, thereby expanding it over the course of the game [3]." Implementing an accurate, inexpensive, and automated system could alleviate the problem of a subjective strike zone that changes over the course of a game. The proposed design determines if the pitch was a strike or a ball without the need for an umpire behind home plate. The current implementation for this application exists, and it uses high performance cameras, inertial measurement units (IMUs), or radar-based systems to record the game. Of these systems, the least expensive system that can be used during gameplay costs $3,000 [4]. There is a need to develop a system that is accurate, inexpensive, and easy-to-use. The system also needs to use a standard baseball and be incorporated into the game without modifying the game itself.

The ability to locate a baseball in 3D space and parametrically estimate the baseball's location follows a general process that can be applied to other fields as well. Object detection and parametric estimation can be used to locate missiles, flying vehicles, or underwater vessels for a military. Staying in the sports domain, object detection can be used in other sports such as basketball, soccer, football, hockey, or cricket. In security, it can be used to track a fleeing perpetrator or a stolen vehicle. There are many applications where the general principles of object detection and parametric estimation apply. A custom, accurate, and inexpensive solution can be applied to these fields as well. Specifically, this project addresses the gap in affordable and accurate technology for a strike/ball detection system.

2. Background

The process to create a smart home plate system that can detect and parametrically estimate a baseball began with investigating currently implemented systems. Once the current systems' pros and cons were classified, multiple sensor technologies were researched to find other feasible implementations for a smart home plate application. This approach led to the consideration of inertial measurement units, radio-frequency identification tags, hall effect sensors, infrared sensors, ultrasonic sensors, and a smartphone camera. Each of these sensors were given a score between 1 and 10 for seven different criteria: detection/position accuracy, baseball modification, affordability, sensor interference, durability, portability, and prototype time constraint. Through a value analysis, the best sensing method was determined for detecting and parametrically estimating fast-moving objects.

2.1 Currently Implemented Systems

High-precision cameras and machine learning techniques are currently used in major league baseball to track a baseball's trajectory in real time, once the baseball is thrown by the pitcher. A convolutional neural network, a popular and accurate machine learning framework, is primarily used for image classification. This framework, along with other image and data processing techniques, is used in unison to produce the real-time trajectory path seen during professional baseball on TV. In addition to high precision cameras, IMUs and radar-based systems have been implemented for this application as well. Examples of currently implemented systems are PITCHf/x, Strike Smart Baseball, PitchTracker, TrackMan, FlightScope Strike, and Rapsodo Baseball.

2.1.1 Camera-based Systems

PITCHf/x

PITCHf/x is a pitch tracking system that was implemented in every professional MLB stadium by 2006 [5]. This product uses three cameras on the field to fully track the baseball's path and the batter's stance. The cameras on the first and third base line are used to judge the trajectory of the baseball and the camera in center field is used to determine the height of the

baseball in relation to the batter's strike zone. Using image recognition, the PITCHf/x system can track the velocity, movement, release point, spin, and pitch locations for every pitch thrown in baseball [6]. The purpose of something this advanced and data-heavy is to analyze and compare the performances of pitchers, and even umpires, at a detailed level.

Despite its main purpose, PITCHf/x was used in a minor league charity event game in 2015 to officially call balls and strikes [5]. An uncommon fifth umpire was used to sit behind the backstop with an interactive display that notified him where the baseball crossed the plate. The umpire would then announce whether the pitch was a ball or a strike. This was mainly done as an experiment to test the accuracy of PITCHf/x in its ability to locate a baseball in real time. Ryan Zander, the Sportvision general manager of baseball products, claimed that the system was accurate to within half an inch. When it was put to the test, the players and the home plate umpire were impressed by the consistency of the called balls and strikes and positively reacted to the system [3]. This trial run showed that there was a desire for more consistency in baseball, but that there is still a gap between accurate and affordable systems.

2.1.2 IMU-based Systems

Strike Smart Baseball

Strike is a startup that is attempting to produce a smart baseball. Their device utilizes gyroscopes and accelerometers, or more compactly, an IMU. Their baseball prototype can collect data on 3D trajectory, spin, speed, rotation axis, and pitch location. This smart baseball design has a small microprocessor with sensors within the baseball itself to collect this information. There is a Bluetooth transmitter in the baseball that transmits all the data to a smartphone application to be viewed. The IMU, transmitter, and other internal components prevent this baseball from being hit. The baseball with an application and wireless charger is projected to retail for $129.99 [7]. The Strike Smart Baseball can be seen in Figure 3 below.

Figure 3: *The Strike Smart Baseball [7]*.

PitchTracker

PitchTracker is a combination of an IMU integrated baseball and analytics tracking smartphone application. The smart baseball provides information about baseball velocity, spin, and timing information. This device allows for more in-depth evaluation about a player's pitching, but it cannot be used in game due to the delicate internal electronics. The product costs $99.99 and requires a subscription to continue using the smartphone application after a two-week free trial [8]. This baseball is very similar to the Strike Smart Baseball mentioned above in terms of visual and electronic design. It can be seen below in Figure 4.

Figure 4: *Diamond Kinetics PitchTracker product [8]*.

2.1.3 Radar-Based Systems

TrackMan

TrackMan baseball is a radar-based system that the MLB decided to use instead of the previous system, PITCHf/x. TrackMan is a high-end system that is rumored to cost around $30,000 dollars [9]. It only has a cost estimate, as the systems are customly installed in MLB stadiums and the price is not advertised [10]. It was believed that TrackMan would result in more accuracy than PITCHf/x, but that was not entirely true. It turned out that both systems had similar errors in determining horizontal baseball location. TrackMan outperformed PITCHf/x in velocity calculations but underperformed in vertical baseball location [10]. The systematic difference between the systems and the defining characteristic of TrackMan was that this system is radar based. It does not use a camera-based approach for object detection and formulates its estimates from thousands of measurements per second, compared to PITCHf/x that uses 20 frames of the baseball during its flight path to formulate its estimates.

TrackMan also uses an analytics system, Statcast, that allows for the collection and analysis of additional data, such as home run distance, average exit velocity, and pitch speed. It also allows you to run a full analysis of a single play. For example, the system can determine that the baseball was hit far into left field with an exit velocity of 100 mph and that the left fielder had a 0.3 second reaction time. It can also determine that an outfielder began to run 0.1 seconds later at an average speed of 17 mph. He reached his max speed of 20 mph before he leaped 3 feet off the ground to rob the hitter of a home run [11]. Statcasts' capabilities of analysis are very advanced and cost the MLB tens of millions of dollars of investment [12]. High-precision camera systems have the capability to analyze and understand many aspects of baseball, but their prices are too expensive for non-professional organizations and recreational users. Below in Figure 5, you can see the TrackMan system using Statcast to display game analytics.

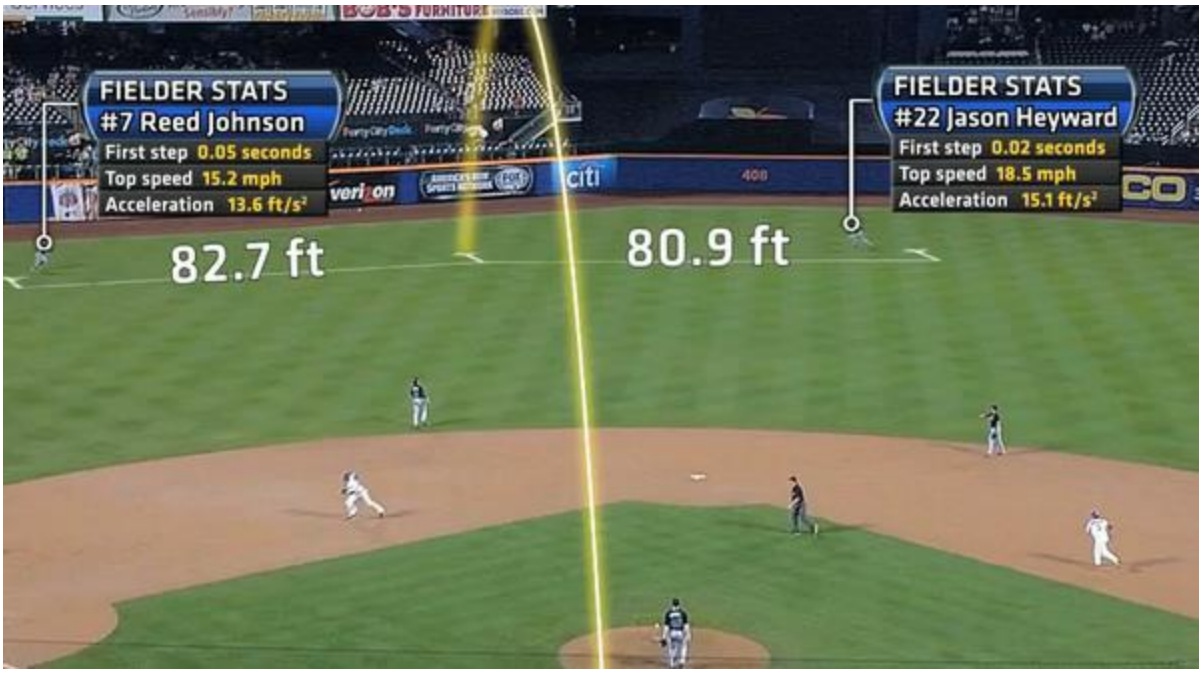

Figure 5*: TrackMan system using Statcast analytics [13]*.

2.1.4 Combined Camera and Radar-based Systems

FlightScope Strike

FlightScope is a hybrid system, originally used in the sport of golf, which is being repurposed for deployment into the sport of baseball. It combines video information with data from the radar system to provide pitch and baseball flight analytics. The data is displayed in real time on a tablet, which is included with the system. The FlightScope system carries a large, one-time cost of $18,000 for the cameras, radar sensors, and tablet [6]. With its high price point, it is comparable to the TrackMan radar-based system. The sensor array, which is about one square foot in area on its face, is shown in Figure 6.

Figure 6: *Flightscope sensor array [14]*.

Rapsodo Baseball

Rapsodo Baseball, seen below in Figure 7 and priced at $3,000, is the most inexpensive system that can also be used during game play. It has the capability to provide several pitching metrics: location, velocity, spin, and the axis of rotation. At $3,000, this system is close to filling the gap in affordability. Since the product outputs these pitching metrics to a tablet or portable computer, access to these devices is required. Although Rapsodo Baseball is much cheaper and able to provide more than just strike/ball classification, this price is not practical for many high schools, colleges, and private leagues.

Figure 7: *Rapsodo system [4]*.

2.2 Value Analysis of Potential Sensors

There are several sensors that could be used to detect fast-moving objects. In order to make the best decision for the sensors, potential options were evaluated through a value analysis comprised of essential criteria. The ideal sensing method used for the design scored the highest in the detailed criteria below:

- **Detection/Position Accuracy**: The ability to accurately report the (x, y, z) position of the desired object. A score of 5 corresponds to a sensor that can locate the object within 1 object diameter of its actual position.
- **Baseball modification**: The degree the sensor option requires modification to the baseball. A score of 1 corresponds to a modification so severe it prevents the use of the baseball in game. A score of 10 corresponds to no modification to the baseball.
- **Affordability**: The ability to implement the final application in comparison to the allotted project budget of $600. A score of 5 corresponds to sensors, which uses the entirety of the available project budget. A score of 1 corresponds to a design which greatly exceeds the available budget, and a score of 10 uses none of the available budget.
- **Sensor interference**: The sensor's level of interference with other sensors in the sensors and the environment. A score of 1 corresponds to complete prevention of other sensors, and a score of 10 corresponds to a sensor which causes no interference within sensors.
- **Durability**: The sensor's likelihood of receiving damage when implemented in the application's use. A score of 1 corresponds to an incredibly vulnerable sensor that is easily damaged, and a 10 corresponds to near immunity to any damage it may receive during a baseball game.
- **Portability**: The ease of setup/transportation of the sensors. A score of 1 corresponds to a non-portable sensor, and a score of 10 corresponds to a sensor which is lightweight and easily deployable.
- **Prototype Time Constraint**: The feasibility of building a functioning sensor prototype within a 7-14-week period. A score of 1 corresponds to a sensing method which exceeds 14 weeks to build, test, and troubleshoot. A score of 10 corresponds to a sensor, which subceeds seven weeks to build, test, and troubleshoot.

2.2.1 Methods Modifying the Baseball

An option for detection was to use sensing methods that physically modified the baseball. This included the addition of transmitters or detectable materials, such as metals, passive sensors, and magnets, to an object to allow for easier sensing. In the following sections, currently implemented baseball-modification solutions were investigated as well as the feasibility of other baseball modification techniques, such as radio-frequency identification (RFID) tags and hall effect sensors. While these solutions have high accuracy, they also modify the baseball to such a degree that it is unusable. These points are outlined in the value analysis in the following subsection.

Inertial Measurement Unit

Embedding an IMU inside a baseball would provide a suitable sensor suite for position and trajectory information to determine a strike or ball. An IMU sensor is mainly a combination of gyroscopes and accelerometers. By measuring the direction of acceleration and changes in movement, it is possible to track an object during its flight path. The baseball would need to be redesigned, as to contain the IMU sensor(s) and maintain the properties that a traditional baseball possesses: center of mass, structural integrity, feel and contour, etc.

Table 2 below shows a value analysis performed on the viability of implementing an IMU sensor system for a Smart Home Plate application. The largest downside was the baseball modification, since this design required the design of a new baseball that could house the IMU sensors. Additionally, this smart baseball was not designed to be hit or used in an actual game of baseball, but instead for practicing, viewing, and analyzing pitch data [7]. Ultimately, the inclusion of an IMU inside the baseball would not be sufficient for the sensor implementation.

Table 2: *Value analysis of IMU for a Smart Home Plate application.*

Criteria	IMU
Detection/Position Accuracy	10
Baseball Modification	**0**
Affordability	8
Sensor Interference	10
Durability	**0**
Portability	9
Prototype Time Constraint	5
Total	**42**

RFID Passive Sensor

Radio-frequency identification (RFID) utilizes electromagnetic fields for local object detection and consists of a reader and a tag. Tags can be active or passive: active tags emit their own radio-frequency (RF) signal detected by the reader, whereas passive tags typically consist of just an antenna excited by the reader and could be placed on a baseball for detection. As active tags typically need a power source, such as a battery, they were not considered for use in an RFID implementation. Alternatively, passive tags use the energy emitted from the reader to send a return signal. As a result, the reader must emit a very powerful signal to effectively power the RFID passive tag. Battery assisted passive tags boost the tag signal when affected by the RFID reader signal, but as previously mentioned, this was not suitable for this application. One advantage of RFID was that it did not need line-of-sight between the reader and tag to function.

Regarding accuracy, RFID tags have two types: read accuracy and location accuracy. Read accuracy is described as the percentage of tags which will successfully be read if they are sent near a RFID reader. Location accuracy is the ability to triangulate the location of a tag from a reader. Most passive tags will only ever notify the reader that they are in the read field, not where. Some more sophisticated systems have real-time location functionality which can triangulate the tag to a few centimeters using ultra-wideband frequency [15]. The NFL already

uses RFID for object and player tracking but sizing this active RFID implementation down to fit a baseball is not realistic [16].

Table 3 below shows a value analysis performed on the viability of implementing a passive RFID sensor system for a Smart Home Plate application. The largest downsides to the RFID option were that it required modification of the baseball, and that the modifications could be damaged if the baseball was struck by a bat. RFID tags could be placed on the surface of the baseball, but this option was also not ideal. With regards to durability, RFID tags contain delicate circuitry and antennae, which could receive enough damage to impair their functionality. RFID passive readers typically fall in the price range of less than $100, where bulk tags are inexpensive at $125 for a roll of 500 tags [17], [18].

Table 3: *Value analysis of passive RFID sensors for a Smart Home Plate application.*

Criteria	**RFID**
Detection/Position Accuracy	9
Baseball Modification	**4**
Affordability	5
Sensor Interference	10
Durability	**2**
Portability	4
Prototype Time Constraint	3
Total	**37**

Hall Effect Sensor

Another method for tracking a moving object while modifying the baseball, was through the implementation of hall effect sensors. At their most basic level, hall effect sensors detect the presence of a magnetic field. Figure 8 below shows how the presence of an object's magnetic field polarizes the metal plate in the hall effect sensor–creating a readable voltage.

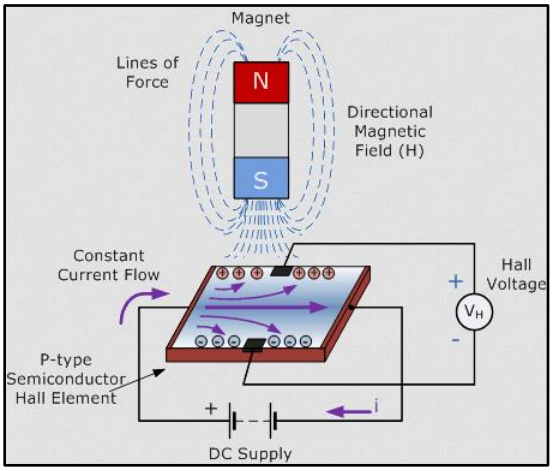

Figure 8: *Diagram of hall effect sensor functionality [19]*.

There were two ways hall effect sensors could be implemented to detect and locate a baseball: coat the baseball internally with magnetic metallic paint or coat the baseball in ferromagnetic metallic paint. To triangulate a magnetic baseball, the resulting voltage caused by the presence of the baseball would be measured at various locations in 3D space around each linear hall effect sensor to generate a model that can predict the baseball's location. If the baseball was a ferromagnetic object, the hall effect sensors always needed to be biased by a fixed magnet, unlike Figure 8 above. This fixed magnet gives the hall effect sensor a specific base voltage value, so it can then detect any variation in the magnetic field caused by the presence of a ferromagnetic object [20]. This method was not viable for triangulation of a baseball because the magnets used on each hall effect sensor would interfere with each other–resulting in inaccurate detection of a ferromagnetic baseball.

Table 4 below shows a value analysis performed on the viability of implementing a hall effect sensor system for a Smart Home Plate application. As mentioned above, sensor interference makes detecting a moving metallic object impossible while other metallic objects or

magnets are already present. Even though potential hall effect sensors were an affordable sensor (cost varying from less than $1 to roughly $100), methods that modify the baseball are unfavorable [21], [22]. Any method that modifies the baseball makes the project's future implementation harder, requiring the design of a new baseball with the appropriate magnetic components.

Table 4: *Value analysis of hall effect sensors for a Smart Home Plate application.*

Criteria	Hall Effect
Detection/Position Accuracy	7
Baseball Modification	**4**
Affordability	7
Sensor Interference	**3**
Durability	7
Portability	4
Prototype Time Constraint	3
Total	**35**

2.2.2 Methods Not Modifying the Baseball

A non-modifying approach would be most ideal, as it avoids tampering with the physical properties of the baseball. The current approach that the MLB takes for autonomously detecting balls and strikes is a camera-based approach, and it uses real-time image recognition [5]. There are several other potential approaches that could determine if the pitch is a strike or ball without modifying the baseball, such as infrared proximity sensors, ultrasonic proximity sensors, or using a smartphone camera as a sensor. The optimal solution is one that can achieve high accuracy, while remaining affordable and minimally disruptive to the game.

Infrared Proximity Sensor

Infrared detection works based on the detection of infrared light from the target object to an infrared photovoltaic sensor. Any object which has a temperature above 0 Kelvin will emit

some degree of infrared radiation. Mechanisms by which the object can be detected are due to any changes in intensity of infrared light, such as passive emission, reflection of actively emitted infrared light, and detection of a shadows cast by an object naturally emitted infrared light. This sensor is less accurate in reporting object distance from the sensor relative to the other sensor options due the nature of the detection medium. One experiment using infrared sensors to identify and resolve airborne, blob-shaped objects found that fast moving objects were unable to be accurately resolved by infrared arrays [23].

Table 5 below shows a value analysis performed on the viability of implementing an infrared sensor system for a Smart Home Plate application. Under perfect test conditions in experiment above, the implementation would not detect a moving baseball at high speeds and distances greater than 30 cm away, so this sensor receives a low score for detection and position accuracy [24]. Also, an array of infrared sensors would likely require several dozen small sensors, which individually cost less than $5 [25].

Table 5: *Value analysis of infrared sensors for a Smart Home Plate application.*

Criteria	**Infrared**
Detection/Position Accuracy	**1**
Baseball Modification	10
Affordability	5
Sensor Interference	5
Durability	3
Portability	7
Prototype Time Constraint	5
Total	**36**

Ultrasonic Proximity Sensor

The ultrasonic proximity sensor sends out and receives sound waves with an ultrasonic transmitter and a receiver. The benefits of ultrasonic sensors are that they have wide angles of detection and small minimum distances for detection. Some commonly used, cost effective

ultrasonic proximity sensors are depicted below with their corresponding prices in Figure 9 and Figure 10.

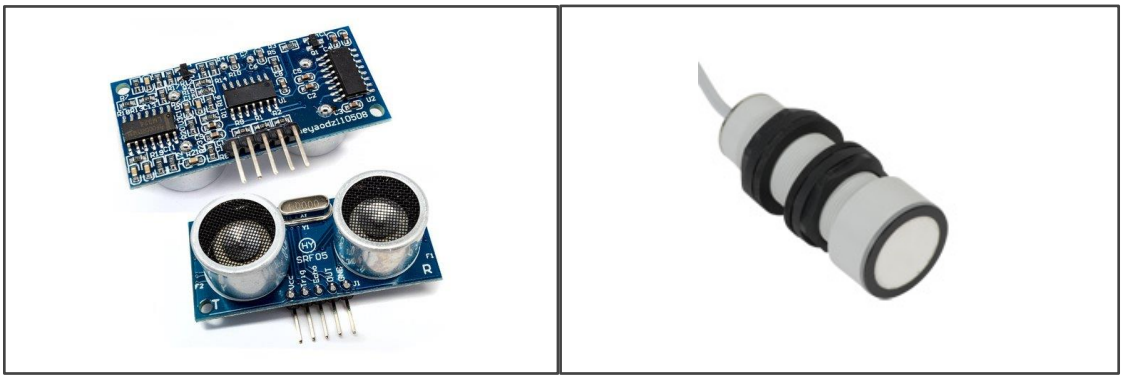

Figure 9: *HC-SR05 Ultrasonic proximity sensor: $6.99 (left) [26]*.

Figure 10**:** *UT2F-EM-0A Ultrasonic proximity sensor: $274 (right) [27]*.

Several factors are used to distinguish the capabilities of ultrasonic proximity sensors. Sensors operating at lower frequencies, such as those near 40 kHz, can detect objects at further distances. Conversely, sensors operating at higher frequencies, such as those near 200 kHz, cannot detect these same objects; their detection range is much shorter.

The feasibility of using multiple ultrasonic sensors to detect and locate fast-moving objects depends on the ability to send and receive multiple signals from multiple transmitters and receivers at once without interference. Frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS) techniques have been implemented in sensor arrays with many signals to deal with discerning between multiple signals. FHSS involves sending out signals at different frequencies at different time intervals whereas DSSS essentially adds unique noise to each signal. Both techniques could theoretically be used to discern between two signals [28].

Table 6 below shows a value analysis performed on the viability of implementing an ultrasonic sensor system for a Smart Home Plate application. The ideal sensors would be able to be covered, as weather, people, and dirt can damage the sensors. This is not a possibility with the ultrasonic sensors, since they need to send ultrasonic signals upward without any obstructions preventing the sound waves from reflecting off the baseball. Secondly, affordability is an issue because the price of a sufficiently accurate ultrasonic sensor such as the one above in Figure 10,

and the budget would be constrained if this this sensor was selected. Therefore, durability, affordability, and time constraint received low scores in the value analysis; however, this sensor excels in its ability to provide a solution that would not modify the baseball

Table 6: *Value analysis of ultrasonic sensors for a Smart Home Plate application.*

Criteria	Ultrasonic
Detection/Position Accuracy	8
Baseball Modification	10
Affordability	5
Sensor Interference	8
Durability	**3**
Portability	7
Prototype Time Constraint	5
Total	**46**

High-Precision Cameras

A value analysis of the high-precision camera-based systems mentioned above in Section 2.1 can be seen below in Table 7. The cheapest system implementation on the market is Rapsodo Baseball, and it costs $3,000. The camera used in that system, as well as the peripheral technology required, would cause a serious budget constraint. This would also result in a system that the consumer is not comfortable purchasing, as its cost is too high. Working with a high-precision camera would not result in a prototype that can be constructed within 14 weeks, as there is a large learning curve related to programming the foreign hardware. Therefore, affordability and prototype time constraint received low scores in the value analysis.

Table 7: *Value analysis of a high-precision camera for a Smart Home Plate application.*

Criteria	High-Precision Camera
Detection/Position Accuracy	9
Baseball Modification	10
Affordability	**1**
Sensor Interference	10
Durability	10
Portability	2
Prototype Time Constraint	**3**
Total	**45**

Smartphone Camera

Detecting fast-moving objects with a smartphone camera was another potential option for the application. Most modern smartphone cameras have slow-motion 240 fps video capabilities. PITCHf/x is an example of using camera-based technology to locate an image. So, the method seems possible; however, the viewable area on a smartphone changes depending on the phone and specific video capture setting selected. Even with equations describing the field-of-view (FOV) of a smartphone camera, the settings alter the result. It also proved to be extremely difficult to determine through camera specifications if a smartphone camera would be able to see the entire strike zone when placed on home plate facing upwards. A controlled experiment was performed to collect the required information to be able to perform a value analysis on this sensor type. With an iPhone 8+ readily accessible, we determined that with the 240fps setting turned on, the bottom of the strike zone was not fully visible. Since a fisheye lens was also easily accessible, we used that to increase the FOV. The resulting image in Figure 11 shows the view of a strike zone as seen through the iPhone 8+ by moving a home plate away from the smartphone in one-foot increments.

Figure 11: *Strike zone through iPhone 8+ camera at 1-foot increments from camera.*

Table 8 below shows a value analysis performed on the viability of implementing a smartphone camera for a Smart Home Plate application. While portability and the prototype time constraint were challenges to be faced, this option excelled in not modifying the baseball since a standard baseball could be used. It exceeded in affordability, as people are assumed to already own a smartphone. It exceeded in sensor interference because the Bluetooth built in on smartphones provides little to no interference. Also, it exceeded in durability and portability, as the phone could be inserted into a customly designed home plate that would also allow the user to remove their phone after use.

Table 8: *Value analysis of a smartphone camera for a Smart Home Plate application.*

Criteria	**Smartphone Camera**
Detection/Position Accuracy	9
Baseball Modification	10
Affordability	10
Sensor Interference	10
Durability	9
Portability	8
Prototype Time Constraint	6
Total	**62**

2.2.3 Value Analysis

After examining the investigated options for detecting fast-moving objects, it was necessary to compare the options and choose which approach best aligned with the process' criteria. The value analysis in Table 9 shows every investigated option's criteria and corresponding score.

Table 9: *Total value analysis of sensor possibilities for a Smart Home Plate application.*

Criteria	IMU	RFID	Hall Effect	Infrared	Ultrasonic	High-Precision Camera	Smartphone Camera
Detection/Position Accuracy	10	9	7	1	8	9	9
Baseball Modification	0	4	4	10	10	10	10
Affordability	8	5	7	5	5	1	10
Sensor Interference	10	10	3	5	8	10	10
Durability	0	2	7	3	3	10	9
Portability	9	4	4	7	7	2	8
Prototype Time Constraint	5	3	3	5	5	3	6
Total	42	37	35	36	46	45	**62**

Based on the value proposition, the ideal solution for a system that can detect fast-moving objects for a smart home plate application is to use a smartphone camera.

3. Methodology

Given the value analysis performed in Section 2.2, this section describes the detailed methodology of our approach to develop a proof-of-concept smartphone-based system for balls and strikes classification. Overall, the system must be designed to process raw video from a smartphone camera and produce a binary output corresponding to a ball or a strike when a pitch is detected.

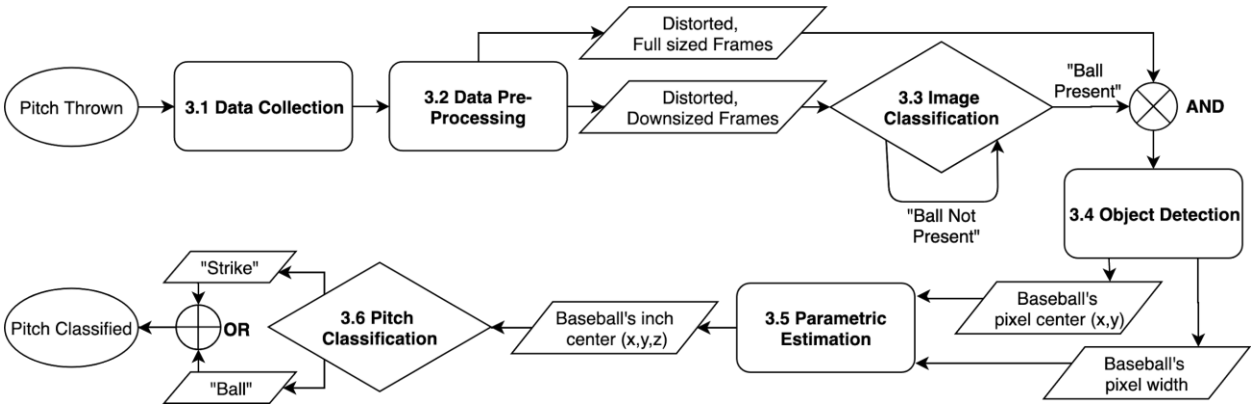

Figure 12: *System flowchart.*

Figure 12 provides an overview of the methods used to detect a pitch, estimate the position and path of the baseball, and classify the pitch as a ball or a strike. A thrown pitch was captured by a continuously filming smartphone with a fisheye lens in slow-motion video. This video goes through a pre-conditioning process to extract every frame of the slow-motion video and prepare the frames for both image classification and object detection. The image classification block utilizes a trained convolutional neural network to classify an image as either "Baseball Not Present" or "Baseball Present." To reduce the computation needed for classification, the extracted frames were pre-processed by making the image grayscale and downsizing the number of pixels. All the image IDs classified as "Baseball Present" are sent into an AND gate with the original distorted, full sized frames. This allowed the object detection block to only receive high resolution frames classified as "Baseball Present." Once the baseball was detected by the object detection block, it outputs the (x, y) coordinates in pixels of the baseball's center point and the baseball's width in pixels. The parametric estimation block accepts these inputs, all in units of pixels, and then outputs an estimation of (x, y, z) coordinates

of the baseball's center point in inches in a coordinate system where the camera lens was the origin at (0,0,0). With this information, the pitch classification block compares the 3D location of the baseball and the 3D volume that makes up the strike zone and concludes that the baseball was either a "Ball" or a "Strike." One of the main contributions of this project was a full proof-of-concept implementation of these steps on a PC. The following sections will delve deeper into how each of these blocks were implemented in the Smart Home Plate system.

3.1 Data Collection

Data collection for image classification was performed to provide training data for an image classification convolutional neural network. The phone that was used for data collection was the iPhone8+ because it met the OS requirement and was capable of filming slow-motion videos at 240 fps at 1080p. The iPhone 8+ could, therefore, be used for future work on a smart home plate application. To gather the pitch data, the phone was used in combination with a fisheye lens placed on the rear camera facing upwards with the screen towards the ground as seen in Figure 13.

Figure 13: *Setup of smartphone with fisheye lens with foam baseball plate for reference.*

The slow-motion video capture feature of the iPhone 8+ was used to obtain video of pitches thrown over the phone. This feature recorded at a frame rate of 240 fps (one frame every 4.2 ms). Data collection took place on WPI's elevated sports field seen below in both Figure 14 and Figure 15. This was done to minimize the number of objects in the image other than the baseball, unlike an indoor sporting area such as a gymnasium. A digital range finding tool was used to determine the phone orientation and placement. The pitcher was set up approximately 50 ft from the phone as seen in Figure 14, which was shorter than the standard MLB pitching distance. This was done for pitching accuracy purposes and to help simplify the data collection process.

Figure 14: *Image of team member facing the phone (left).*

Figure 15: *Image showing the field the data collection took place on (right).*

The data collection had to be done during different times of day facing different directions to prevent shadows from biasing the data used to in the convolutional neural network dataset. Many sessions were conducted to ensure enough frames to train the image classification network were collected.

3.2 Data Pre-processing

To process these individual frames, VideoLoupe, a 3rd party video processing program, was used to perform frame extraction, a process which individually separates all the images of a video to prepare them for image processing. Images were extracted using a mono filter to reduce

the amount of information they carried by a factor of three. Instead of an intensity value for each red, green, and blue pixel, there was now only one light intensity value from 0 (black) to 255 (white) seen in Figure 16.

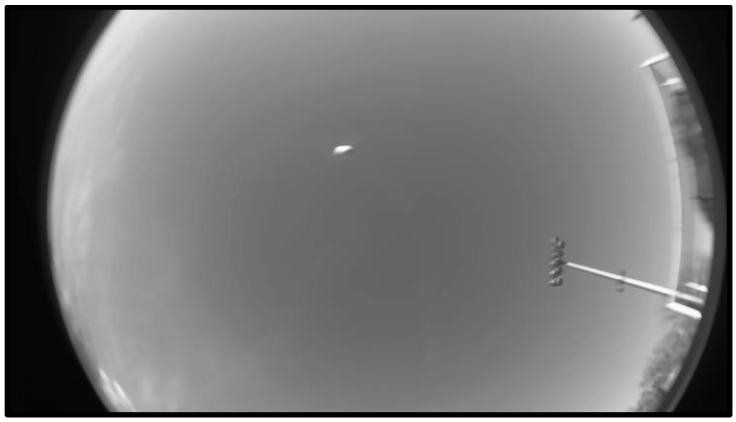

Figure 16: *Sample image after frame extraction and mono conversion*.

The images were kept in chronological order with respect to their source video for purposes which will be further explained in the methodology. The mono images were then separated into a training set and a test set for the convolutional neural network. A total of 600 images were manually processed before being fed into a convolutional neural network. Half of the dataset consisted of images without a baseball, and the other half consisted of images with a baseball present. This 600-image dataset was split into 480 images for training the neural network, and 120 for testing. The next data-preprocessing step for Figure 16 above would be to downsize and reshape the image, shown in Figure 17, for input into the image classification network. The image in Figure 17 appears to be distorted because the aspect ratio changed from 16:9 (or 1920x1080 pixels) down to 1:1 (or 200x200 pixels). It was critical to preprocess the frames to be this small to allow for desired computation speed from the convolutional neural network block.

Figure 17: *A downsized image for input into the image classification neural network.*

3.3 Image Classification

The next step was to determine which processed frames contain a baseball. This involved sending individual frames through a tool that outputs the frame's class: "Baseball Present" or "Baseball Not Present." The tool used to determine these outputs was a convolutional neural network (CNN).

A CNN, often used for image classification as they produce results with high accuracy, was constructed for determining whether there was a baseball in a specific image or whether there was nothing but the background. The CNN has a general structure seen in Figure 18, with the dimensions of a single image as it goes through the network located above the flowchart.

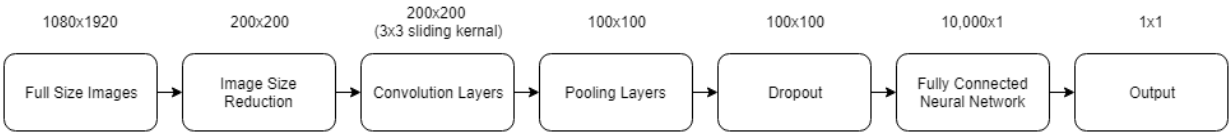

Figure 18: *CNN flowchart.*

The input layer consisted of a 200x200 array of integers from 0 to 255 corresponding to the monochrome images described in Section 3.2. Next there was a convolutional layer that utilized 16 different kernels (3 x 3 grouping of pixels that slides over the image to locate the desired qualities that map to an appropriate output). Next, a pooling layer was used to further

reduce the dimensions of the image, providing an even less computationally intense array for the fully connected neural network. Next, there was a dropout layer, which was a regularization technique that prevented models from overfitting. Specifically, as the model became more complex, the variance increased. This meant it became very sensitive to new data, and this regularization technique helped reduce variance and ultimately produced better results. Finally, the dimensionally reduced and predictive data was sent through dense layers (a fully connected neural network). The job of the fully connected neural network was to map the inputted pixel values to either a 1 or 0 where a 1 represented a baseball being present in the image and a 0 represented a baseball not being present in the image. This was a supervised learning problem, as the outputs were known and used during the training. The training consisted of sending the images through this network many times, as the kernels mentioned earlier were adjusted each time to produce kernels that accurately assisted the CNN to produce the appropriate output.

To implement the CNN in Python, the machine learning package Keras was used with the TensorFlow package as a backend. A Keras Sequential Model was implemented. Additional layers can be added to this model to create the full flowchart in Figure 18. For the specific CNN used, a Sequential object was created and called "model." Then each component mentioned above was added to the model using "model.add(component)." This full implementation in Python can be seen in the code snippet in Figure 19 below.

```
model = Sequential()
model.add(keras.layers.Conv2D(filters, (num_conv, num_conv),
                    strides=(1,1),
                       padding='valid',
                        input_shape=(img_rows,img_cols,1),
                             data_format="channels_last"))
convout1 = Activation('relu')
model.add(convout1)
model.add(keras.layers.Conv2D(filters, (num_conv, num_conv)))
convout2 = Activation('relu')
model.add(convout2)
model.add(MaxPooling2D(pool_size=(num_pool, num_pool)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
model.load_weights(weight_file)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 19: *CNN implemented in Python.*

In total, 480 training samples were used to create a trained network, and in order to determine if the system accomplished this task with high accuracy, test data was run through the network and an accuracy score was outputted. This score described how many frames it accurately categorized over the total amount of frames it examined. A testing accuracy of 89.17% was achieved when predicting on 120 test images.

3.4 Object Detection

This section explains the required steps taken after an image was classified as a having a baseball present. The flowchart in Figure 20 shows the steps executed on images with a ball present to identify the location and size of a baseball in the image. As shown in the system flowchart in Figure 12, this step outputted the baseball width and baseball center coordinates in pixels.

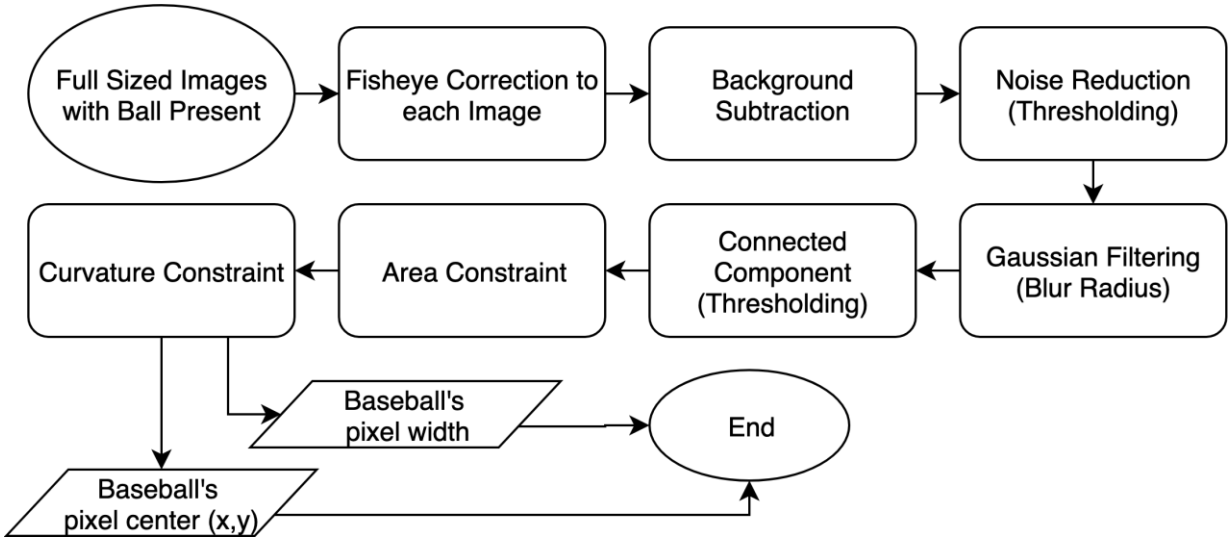

Figure *20: Object detection flowchart.*

Once the neural network successfully classified images containing a baseball, the images were then sent through a mathematical model to detect the baseball's location in 3D space. This task had to utilize many image conditioning techniques to differentiate the baseball from all other objects in the given image such as fisheye lens correction, noise subtraction, and filtering. We knew from experimentation that a typical 75 mph fastball would consist of 30 frames containing the baseball. In a future product implementing this process, once an image was classified as

containing a baseball, the previous 15 images and the next 45 images would be stored in an image buffer. This would guarantee that the entire baseball path and a background image were captured. This was a critical step to ensure the first image in the buffer of 60 images did not have a baseball and could be stored as the reference image for the specific pitch, enabling us to subtract the reference image from the image with a baseball. This buffer was simulated by manually creating a buffer and importing the images into a python environment. Before the subtraction was performed, a fisheye lens correction algorithm was applied to all 60 images in the buffer. This method was done so it would be easier to characterize the 3D coordinate location of the baseball in the next objective. An undistort fisheye function from the OpenCV library, which output weights specific to the fisheye lens to correct a distorted image, was implemented in Python. Applying these weights to the image took the original, circularly distorted image and modified it to look like a normal, square image. The 60 fisheye-corrected images were further processed to determine where the baseball was located in the image. In Figure 21 below, a distorted line appears curved before fisheye correction (shown top), and a straight undistorted line was outputted (shown bottom) after fisheye correction. The color difference between the distorted and undistorted images was not an artifact of the fisheye correction function. The images in Figure 21 were displayed in different plotting environments, so the color differences can be ignored.

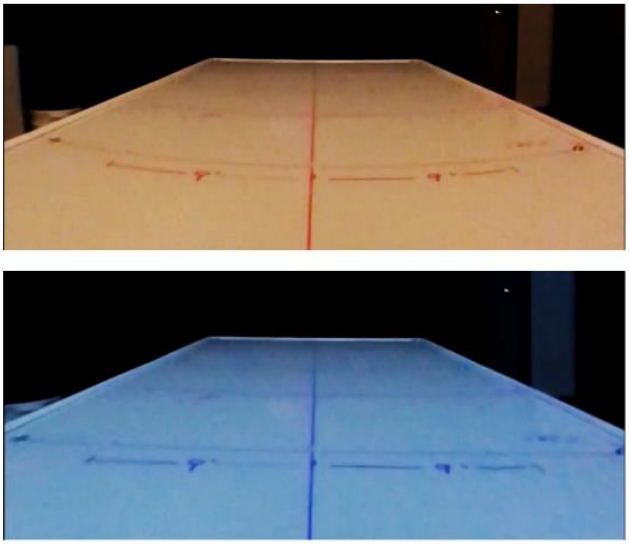

Figure 21: *Distortion versus undistortion with fisheye correction.*

A grayscale image contains pixels whose values range from 0 to 255 where 0 represents the color black, 127 represents gray, and 255 represents white. Therefore when taking one 1920x1080 grayscale image and subtracting another 1920x1080 grayscale image from it, the result was a new 1920x1080 whose pixel values accentuate the differences between the two images and zero out the similarities. Next, an image containing a baseball was subtracted from the reference background image to isolate the baseball's pixel location and blackout the other pixels. Due to noise, there are several other connected pixels resulting from the image subtraction. More filtering methods were then used to reduce this noise and remove the remaining connected pixels which were present after image subtraction.

Filter 1: noise reduction (thresholds)

Non-idealities are present after the subtraction; some pixels are just above and below zero, resulting in pixels with values [0-5] and [250-255]. To eliminate this noise, all pixels with values [0-5] and [250-255] were set to 0.

Filter 2: Gaussian Filtering (Blur radius) to smooth image and remove small objects

There are still non-idealities present in the image due to the subtraction and imperfect noise reduction filter, so a Gaussian blur filter was implemented to smooth the conditioned image and remove small objects.

Filter 3: (ndimg.gaussian_filter object from scipy package) Connected components

Using the resulting blurred image, the ndimg.gaussian_filter object from scipy package was used to classify clustered pixels as an object. The number of objects produced by the scipy package was determined by a threshold value: a low threshold results in few, very large objects and a high threshold results in many small objects.

Filter 4: Iterate through objects and filter by area

Iterating through all found objects in the image, check to see if the object fits within a predetermined area constraint that represents a baseball area between (30x30) and (350x350).

Filter 5: iterate through objects and filter by curvature

Iterating through all found objects that passed the area constraint, find the upper edge of the current object. Then, the upper edge of the found object was compared with a reference curve that represents the curvature of a baseball through a mean squared error (MSE) analysis. The object with the smallest MSE was outputted as the baseball given that the value was below a determined threshold of 10,000. After this filtering, the location and size of the pixel cluster

representing the baseball was extracted as the (x, y) pixel location of the center and the baseball's width in pixels. These filtering methods can be visualized in Figures 32 and 33 in Section 4.4 of the results.

3.5 Parametric Estimation

To determine if the full strike zone falls within the smartphone camera FOV with a fisheye lens, the controlled experiment mentioned in Section 2.2.2 was performed. The relationship of how the width of an object in an image was a function of the object's distance from the smartphone was formulated from this preliminary controlled experiment. Since the plate width and height are exactly 17 inches, the relationship between apparent size of an object and the object's distance from the camera can be calculated. The relationship can be seen in Table 10 and Figure 22 below.

Table 10: *Distance from camera in inches and pixel width of the object.*

Inches Away	Pixels
12	1044
18	761
24	590
36	400
48	301
60	243
66	223

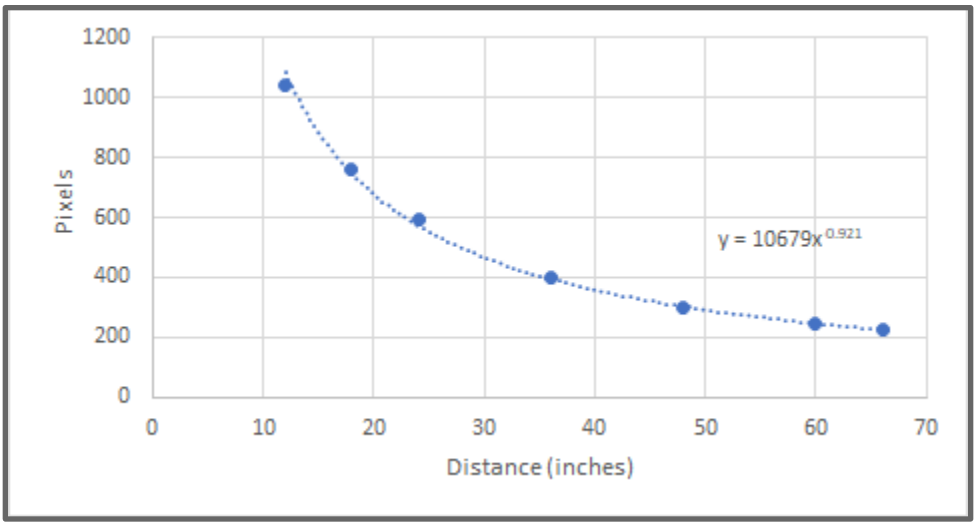

Figure 22: *Pixels vs. Distance in inches.*

Given only the baseball's pixel (x, y) center position and pixel width, the baseball's (x, y, z) location from the camera lens had to be accurately determined in the inches. This meant a solution was needed to map pixel width and (x, y) coordinates in an image to (x, y, z) (in inches) in 3D space, where the camera lens was located at (0, 0, 0). Figure 23 below shows the strike zone with the defined x, y and z axis. The black dot represents the origin and camera lens located at (0,0,0). This was done with the limited information extracted after the filtering stages in the "Object Detection" block. To characterize (x, y, z) in inches, an experiment was conducted in which the baseball's (x, y, z) locations in pixels and inches, and the baseball's width in pixels were known. This was accomplished by setting up the camera in a fixed position perpendicular to a white board table as seen below in Figure 24.

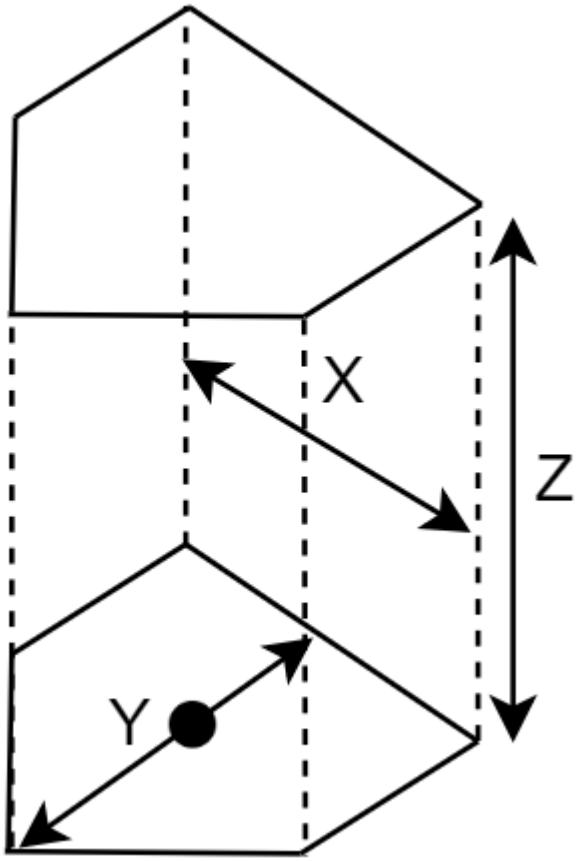

Figure 23*: Strike zone volume with defined axes.*

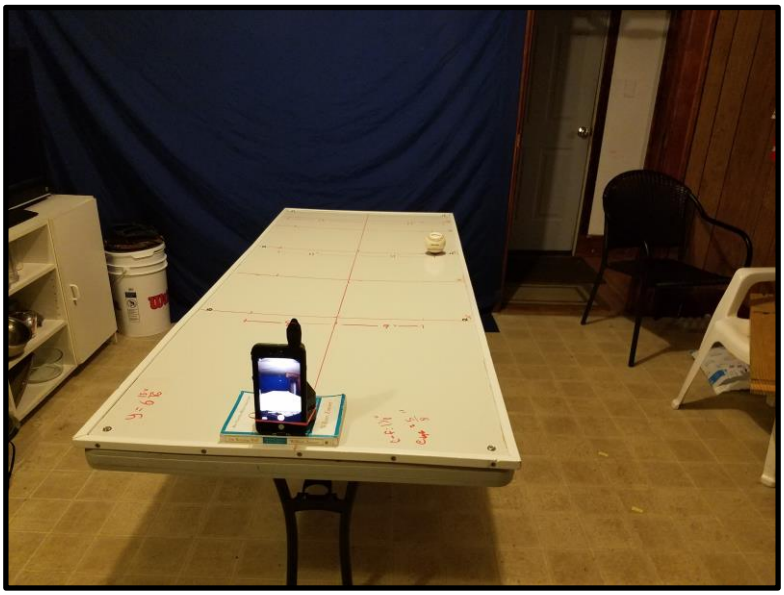

Figure 24: *Setup of validation experiment on table.*

Once the camera was set up, a baseball was placed on the table in pre-measured (x, y, z) locations in reference to the camera lens and then recorded via slow-motion video. Recording the still baseball in slow-motion video was required to ensure the resolution and distortion were consistent with the system's end use scenario. A total sample size of 20 baseballs in unique (x, y, z) locations were recorded for this controlled experiment where 15 baseballs were in different locations on the table and 5 baseballs were suspended from the ceiling. Each baseball's location was stored in Table 11 seen below.

Table 11: *(x, y, z) and calculated d from controlled experiment.*

	Actual[x,y,z,d] From Experiment			
img_ID_by_order	**x**	**y**	**z**	**calc, D**
1	9	5.3	24	26.17
2	9	5.3	36	37.48
3	11	5.3	48	49.53
4	11	5.3	60	61.23
5	13	5.3	66	67.48
6	0	5.3	24	24.58
7	0	5.3	36	36.39
8	0	5.3	48	48.29
9	0	5.3	60	60.23
10	0	5.3	66	66.21
11	-9	5.3	24	26.17
12	-9	5.3	36	37.48
13	-11	5.3	48	49.53
14	-11	5.3	60	61.23
15	-13	5.3	66	67.48
16	3.125	-23.3875	53.6875	58.64
17	10.875	-23.3875	26.125	36.71
18	-6.875	-23.3875	34.5	42.24
19	-12.375	-23.3875	49.5	56.13
20	-4.4375	-23.0125	69.125	72.99

The next step of this controlled experiment was to extract a frame per baseball position out of the slow-motion video, undistort that image with the fisheye correction script, and then manually estimate the (x, y) pixel location of the baseball's center point as well as its width in

pixels, thus producing the (X_P, Y_P, W_P) vector used as an input to the parametric estimation function. This manual image processing was completed with GIMP, a program used for image editing. This step was essential to create a control for the expected outputs from the "Object Detection" block's filtering stages. The manually measured pixel (x, y) locations were normalized to reflect an origin located in the center of the image (540,960) and the widths were collected and stored in Table 12 seen below. After the experimental data was collected, various regression models were analyzed with the inputs of pixel widths and center (x, y) pixel locations of the baseballs to determine each baseball's (x, y, z) location in inches.

Table 12: *Parameters from manual GIMP measurements (pixels).*

Measured W	Measured X_P	Measured Y_P	Normalized X_P	Normalized Y_P
131	906	1113	366	153
79	779	1027	239	67
61	754	988	214	28
48	711	965	171	5
46	723	955	183	-5
118	532	1105	-8	145
75	538	1026	-2	66
56	540	988	1	28
47	542	964	2	4
44	543	955	3	-5
121	182	1096	-358	136
73	310	1020	-230	60
55	338	981	-202	21
47	376	959	-164	-1
45	366	950	-174	-10
53	597	484	57	-476
111	895	52	355	-908
83	376	267	-164	-693
57	341	459	-199	-501
41	487	582	-53	-378

Determining x position in inches

In order to determine the x position in inches, the x position in pixels and the baseballs width in pixels were used. The relationship between the inputs and the response variable can be

seen in equation (1) where X_I was the x position from the camera lens in inches, X_P was the x position away from the normalized pixel origin in pixels, and W was the width of the baseball in pixels.

$$X_I = (3.1895 \cdot \frac{X_P}{W_P}) - 0.0474 \quad (1)$$

This model achieved an adjusted R square value of 0.996 when predicting X_I values and comparing them to the control experiment's actual x position in inches. The differences between the actual x position in inches and the predicted x position in inches can be visualized when viewing the histogram of errors in Figure 25 below.

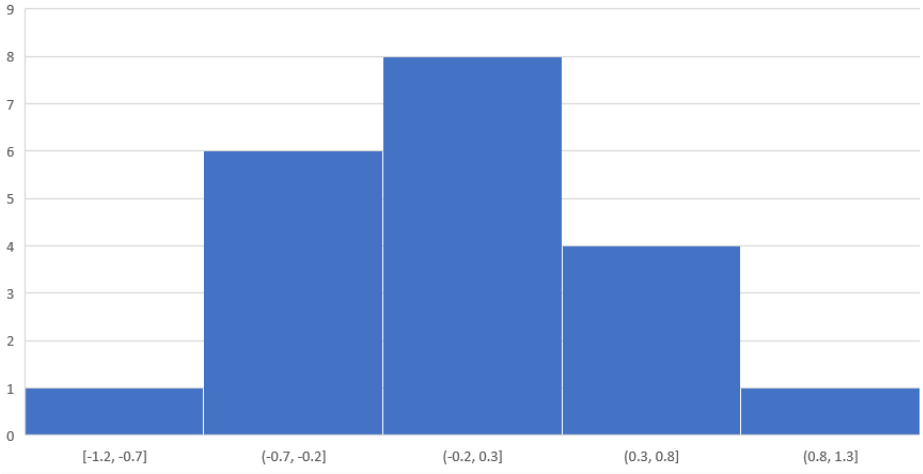

Figure 25: *Histogram of errors for x in inches.*

Determining y position in inches

In order to determine the y position in inches, the y position in pixels and the baseballs width in pixels were used to characterize Y_I in equation (2) below where Y_I was the y position from the camera lens in inches, Y_P was the y position away from the normalized pixel origin in pixels, and W was the width of the baseball in pixels.

$$Y_I = (3.0739 \cdot \frac{Y_P}{W_P}) + 3.7391 \quad (2)$$

This model achieved an adjusted R square value of 0.985 when predicting Y_I values and comparing them to the controlled experiment's actual y position in inches. The differences between the actual y position in inches and the predicted y position in inches can be visualized when viewing the histogram of errors in Figure 26 below. The errors in the y direction were larger than the errors in the x direction. This was due to keeping the y direction constant for most of the collected data. There were not enough changes in the y direction for the formulas to best fit the location of the baseball in the y direction. There also appears to be a bimodal distribution, rather than a normal distribution, which can be explained by the lack of a large sample size. With this in mind, more controlled data points can be collected to fit an even better model to predict the y coordinate in 3D space.

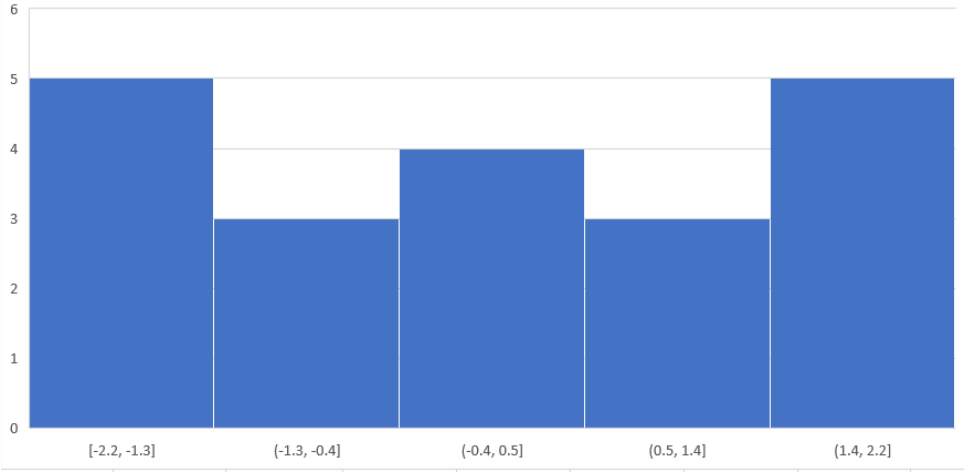

Figure 26: *Histogram of errors for y in inches.*

Determining z position in inches

In order to determine the z position in inches, the x and y position in inches and the baseballs width in pixels were used to characterize Z_I in equation (3) where X_I was the x position from the camera lens in inches, Y_I was the y position from the camera lens in inches, Z_I was the z position from the camera lens in inches, and W was the width of the baseball in pixels. Using X_P and Y_P as features instead of X_I and Y_I proved to not improve the model's performance, so X_I and Y_I were used.

$$Z_I = (0.1043 \cdot X_I) - (0.0074 \cdot Y_I) + (2897.7004 \cdot \frac{1}{W_P}) - 0.8009 \quad (3)$$

This model achieved an adjusted R square value of 0.986 when predicting and comparing Z_I values to actual z-position in inches measured from the controlled experiment. The differences between the actual z position in inches and the predicted z position in inches can be visualized when viewing the histogram in Figure 27 below. It was expected that the greatest error would be found when predicting z because z suffers from compounding errors incurred by its dependence on x and y.

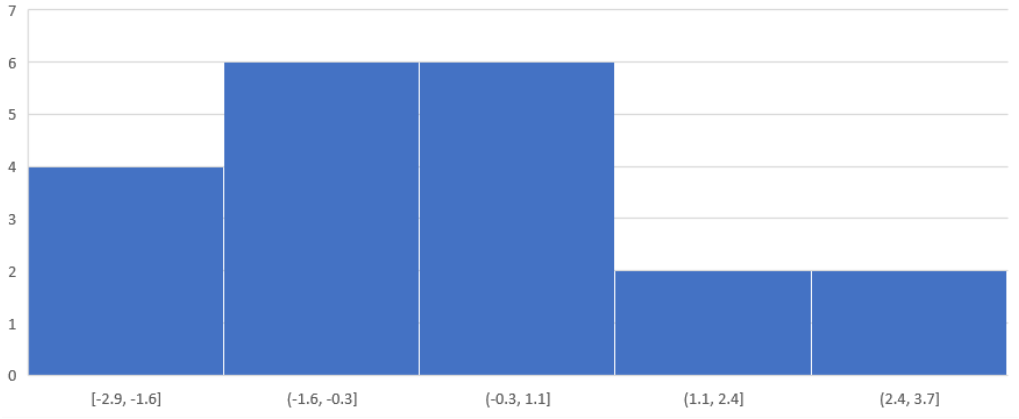

Figure 27: *Histogram of errors for z in inches.*

3.6 Pitch Classification

After the (X_I, Y_I, Z_I) location of each baseball was known, a line of best fit was approximated using principal component analysis (PCA), a mathematical process that can be used to reduce the dimensionality of the data. Specifically, the first principal component was a vector pointing in the direction of max variance in the data. With a baseball travelling over the home plate, the direction of max variance was the direction from the pitcher towards the catcher. The estimated baseballs now located in 3D space were projected onto this best fit line. In addition, many more points were projected onto this line, so that the best fit line appeared less discretized.

All of the points projected onto the vector, were checked to see if they were located within the strike zone volume. (Please refer to Figure 23 in Section 3.5 for the following process. The arrows can represent the bounds of each axis that lie within the strike zone volume.) In order to perform this check for each baseball, it was first checked if the baseball's z coordinate was

located within the z bounds. If true, it was checked if the baseball's x coordinate was within the x bounds. Lastly, it was checked if the baseball's y coordinate was located within the y bounds. If all were true for any baseball along the best fit line, the pitch was classified as a strike. If any were false for all baseballs along the best fit line, the pitch was classified as a ball.

4.0 Results

The results section outlines the performance of the project process, using a single pitch from the collected data dataset. A single extracted pitch from the high-frame-rate video was used to verify all the steps in the methodology. Therefore, this section mirrors the methodology, following the six steps outlined in the system flow chart: data collection, data pre-processing, image classification, object detection, parametric estimation, and pitch classification.

4.1 Data Collection

On the WPI rooftop field, the MQP team pitched 46 pitches (seen in Figure 28 on the left), and in the WPI gym, another 50 pitches were thrown (seen in Figure 28 on the right). In total, we pitched 96 pitches. The pitches were filmed with the pitching apparatus seen above in Figure 13. Filming at 240fps, the baseball was seen by the camera for about 25 frames per pitch. Taking about a ten second break between each pitch resulted in an additional 1,200 empty frames per pitch. With 96 pitches and ten seconds between each pitch, there resulted in a total of about 116,400 frames of data.

Figure 28: *Pitching for data collection.*

4.2 Data Pre-Processing

The frames for a pitch example were extracted using VideoLoupe. The frame extraction process can be seen in Figure 29 below. The pitch frames were added to their own dataset and were used to validate the process established in the Methodology Section. The background image

was extracted separately and labeled. This process involved the manual extraction of several pitches from the dataset of about 116,400 frames, but in a final product the CNN would label and aggregate these pitches automatically. Manual extraction was used to simplify the process of validating the data pre-processing methodology.

Figure 29: *VideoLoupe frame extraction.*

4.3 Image Classification

Once the data pre-processing was complete, the down sampled distorted images were sent through the CNN. The output of the CNN was 1 if "Baseball Present" in an image and 0 if "Baseball Not Present." The CNN also kept track of which frames were classified as which class by displaying the "Corresponding IDs." The output seen in Figure 30 shows 18 ones and 3 zeros. All these frames contained the baseball, but the CNN was only able to classify 18/21, approximately 86%, of the images correctly. This aligned with the previously mentioned testing accuracy of approximately 90% when predicting on the full testing set, not just a single pitch. You can see in this example that images with IDs 4, 10, and 15 were predicted incorrectly, as those images were predicted as 0 instead of 1.

```
Prediction:  [1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1]
Corresponding IDs:  ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21']
```

Figure 30: *CNN output.*

Figure 31 below shows how the image classification process starts with a full-size undistorted image, downscales the image, and send it through the CNN to be predicted. When in the CNN, convolution extracts the important features, the image is downscaled again with pooling layers and the new pixel values are sent through the multi-layer perceptron (MLP) feed-forward neural network. The CNN then outputs that the baseball was present for this frame example.

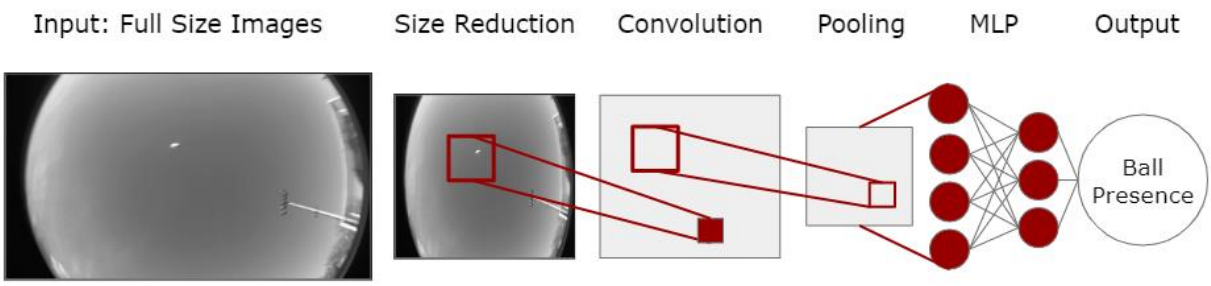

Figure 31: *CNN process-flow diagram.*

4.4 Object Detection

Once the distorted downsized frames were classified, the IDs of the "Baseball Present" frames were used to retrieve the same frame from the distorted full-size frames. These full-size frames with the baseball were processed through the Object Detection Flowchart, shown above in Figure 31, where the images had to be undistorted through fisheye correction, subtracted from a background image, and then filtered to reduce noise. Below in Figure 32, a full-sized distorted background image can be seen on the left. This distorted image in addition to a full-sized image with a baseball present undergoes fisheye correction to become undistorted, as seen in the middle of Figure 32. Now that both the background frame and frame with a baseball were undistorted, the background image was subtracted from the image with the baseball. This resulted in the image on the far right in Figure 32. This step was essential to accentuate the baseball in the image.

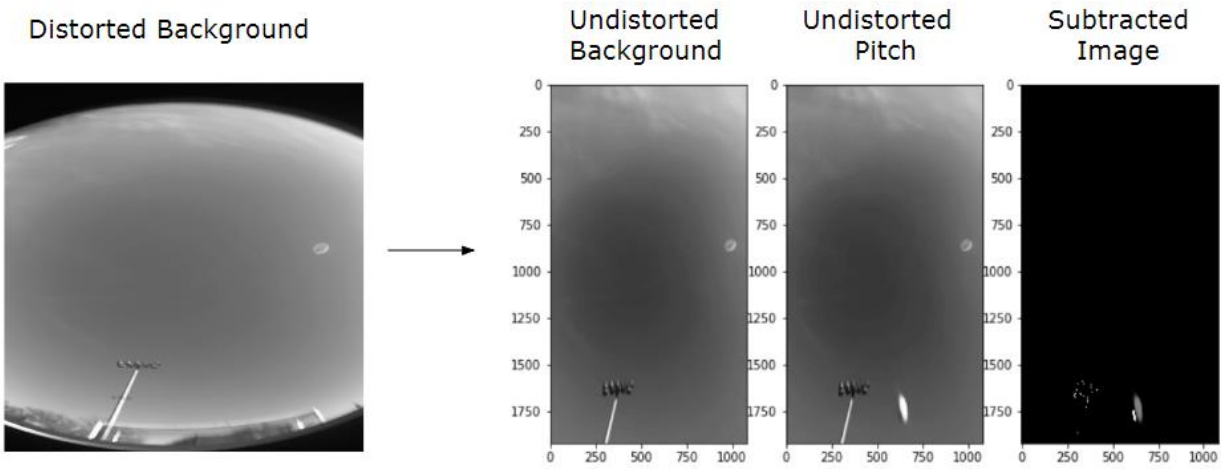

Figure 32: *Data collection steps of fisheye correction and subtraction.*

Once a subtracted image was created, the image was filtered using a thresholding filter and a Gaussian filter to reduce the present noise. Once filtered, the image dimensions were squared off, allowing the connected components script to run. This script detected 66 objects in the subtracted image as seen below in Figure 33 on the left. After applying the area constraint filter, only 6 objects remained in the image that could potentially be the baseball.

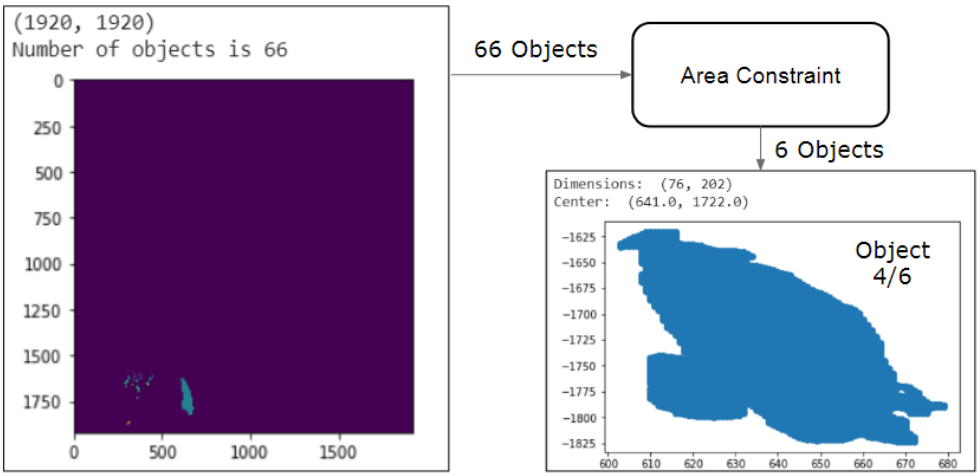

Figure 33: *Data collection steps of connected components and applied area constraint.*

The remaining 6 potential objects were then analyzed through the curvature constraint filter. The visualization of the curvature constraint applied to object 4 out of 6 can be seen below in Figure 34. A curve was collected from the object by plotting the minimum values of each

column of the image. Then, a best fit line of that curve, depicted in orange below, was generated. This orange curve was the aligned with an empirically derived reference curve, depicted in green below, in order to perform a mean-squared-error comparison. Object 4 out of 6 had the lowest mean-squared-error value; therefore, object 4 must be the baseball. Object 4's (x, y) pixel location and pixel width were recorded and then used for parametric estimation.

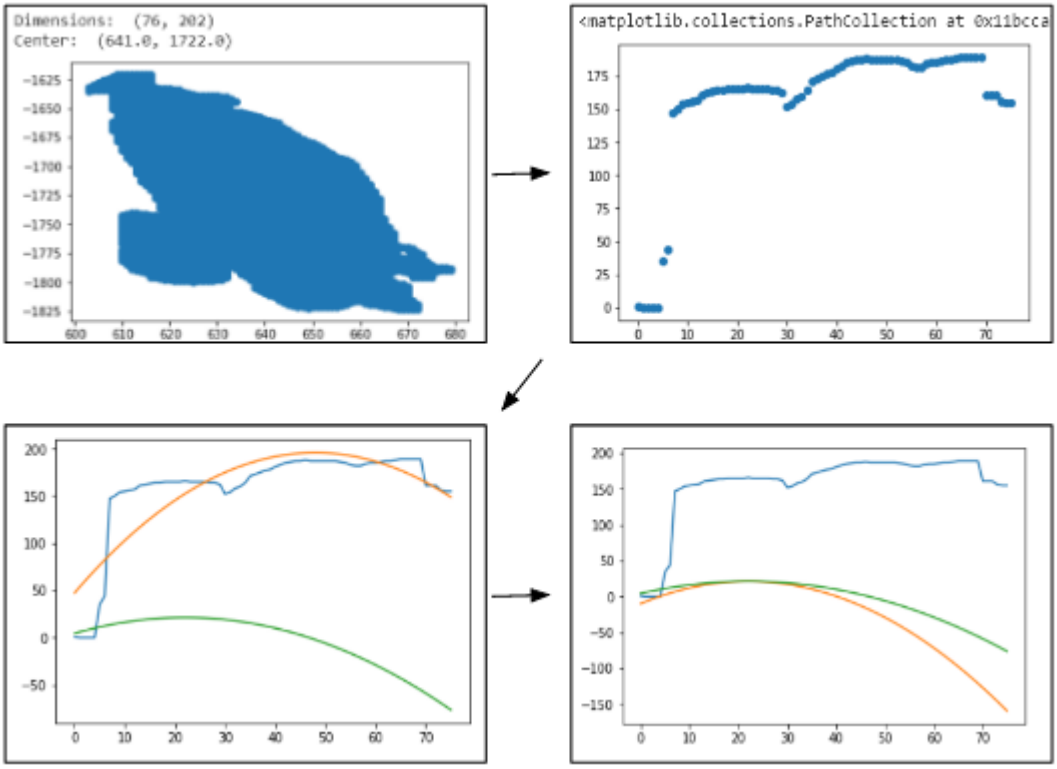

Figure 34: *Object detection curvature constraint process visualization.*

4.5 Parametric Estimation

From there, the equations in Section 3.5 are used to determine the (x, y, z) center location in inches relative to the origin (the camera located at the center of the plate). The determined (x, y, z) center location for the "Baseball Present" outputs from the CNN above are shown in Table 13 below. As you can see, there were 18 baseballs present as there are 18 "1" images in Figure 30 above, and there are 18 rows in Table 13.

Table 13: *Output of parametric estimation section for pitch example.*

X	Y	Z
11.913251	-15.511373	72.999130
13.130304	-11.832781	76.911476
13.466043	-6.776989	76.908936
13.569346	-1.778237	74.927322
14.060041	3.305555	74.940727
13.470039	12.851037	69.501160
12.046147	16.002720	60.705162
13.100460	21.567799	64.798347
13.525728	26.451916	64.806419
12.391679	28.514862	58.233505
-13.225183	39.682362	48.362076
-10.046324	43.675632	43.822304
-10.046324	43.675632	43.822304
12.828760	44.895443	53.864532
11.464725	46.629860	45.324856
-21.649172	65.496925	84.264137
-21.649172	65.496925	84.264137
-22.473734	67.426857	86.907845

Outliers are removed from this data based on their deviation from the median of the returned data. A threshold value was of 95% was empirically derived, meaning that if a baseball was larger 1.95 times the median or less than 0.05 times the median in either the x or z direction, the baseball was eliminated and labeled as an outlier. A larger threshold of 99% was used for the y direction because the baseballs path should not deviate by much in the y direction, as that was the direction of travel for a pitched baseball. Therefore, this larger threshold combined with the two slightly smaller thresholds ensure that any frame incorrectly detected as a baseball was removed. The full path of the baseball can be visualized in 3D space, as seen in Figure 35.

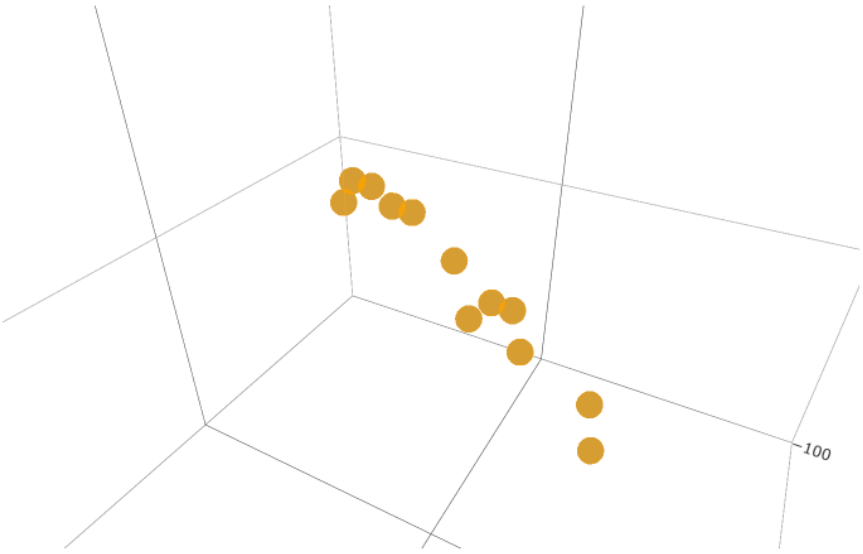

Figure 35: *Extrapolated (x, y, z) baseball path.*

4.6 Pitch Classification

The best fit line for the baseball path was determined and plotted in 3D space. In order to calculate the best fit line of 3D points in a 3D space, a simple linear regression would not work, as it would produce a plane in 3D space, not a line. The dimensionality of the data needed to be reduced, so a common mathematical technique, PCA, was used. The technique allows us to extract the vectors that correspond to the most variance in the data. The vector that corresponds to the direction of most variance in this case was the first principal component and was also the best fit line in 3D space. This best fit line can be seen in Figure 36 below.

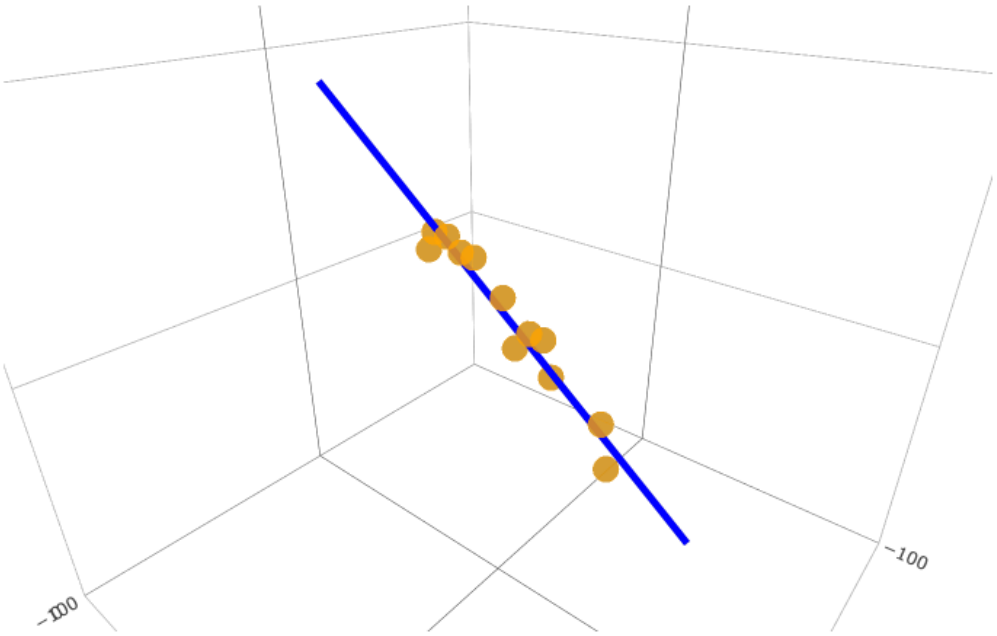

Figure 36: *Best fit line of baseball path generated through PCA.*

Using the best fit line as the determined path of the baseball, the next step was to determine if the line ever intersects the strike zone volume. If it intersects the strike zone volume, the pitch was a strike, and if it did not, the pitch was a ball. It was only possible for the baseball to intersect the volume if it crosses the plane making up the front of the strike zone, the plane making up the right of the strike zone, or the plane making up the top of the strike zone. Those planes either have a constant x coordinate (the side planes), a constant y coordinate (the front plane), or a constant z coordinate (the top plane). Utilizing this attribute, you can see if a particular plane was crossed by checking if the baseball was on the one side of the constant dimension that was within the strike zone and within the bounds of the plane in the other two dimensions. The proximity of the pitch in reference to the strike zone can be seen in Figure 37 below.

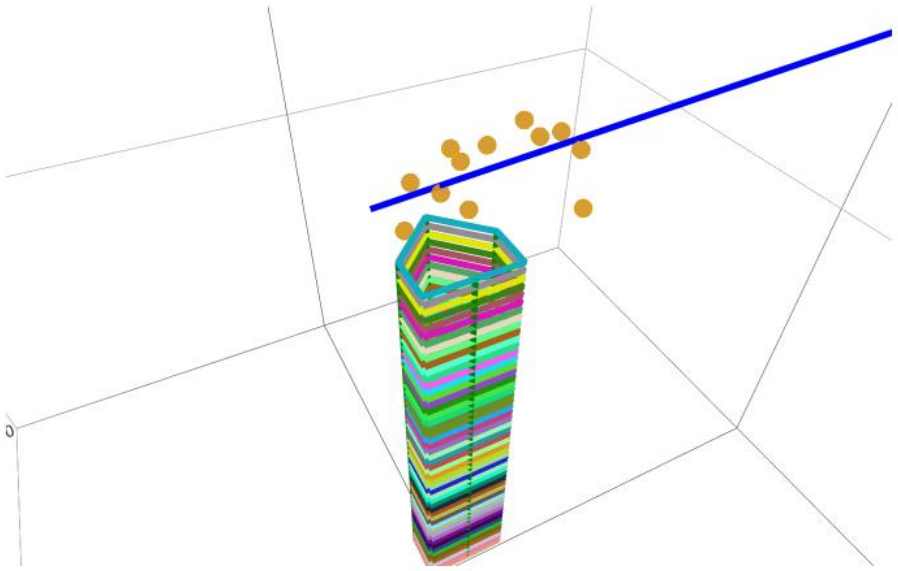

Figure 37: *Baseball path and strike zone (showing that pitch was high and outside).*

Finally, the constraints are checked using a series of nested if statements and they are checked for approximately 1000 points on this line to better check if any point on the line truly intersects the strike zone. The line is passed into the “IsStrike” function and an output is returned, as seen in Figure 38. In this case, the pitch can be classified as a ball and you can go back and examine how close it was to being a strike.

```
#Check if any point on line crosses strike zone
results = isStrike(pc1_line)
print("Is Strike: " + str(np.array(results).any()))
```

```
Is Strike: False
```

Figure 38: *Final pitch classified.*

5.0 Future Work

While this project successfully developed a proof-of-concept for an inexpensive smartphone-based system for calling balls and strikes in a baseball game, there are several additional steps that should be carried out to produce a final product that can detect balls/strikes in real-time. These steps include developing a more robust baseball detector by collecting more pitch images in real and synthetic environments with different backgrounds. These steps also include developing enhanced filtering methods for faster and more accurate real-time object detection, and a smartphone application to run all the steps described in the methodology in real-time on an iPhone 8+ or a phone with similar/better camera specifications.

5.1 Improving Baseball Detection

As mentioned in Section 4.3, the CNN used for baseball detection had a test accuracy of about 90%. However, this CNN was only trained and tested on a single background. To improve the detection accuracy and make the detection robust to more challenging environments, a CNN must be trained to not only classify images with the background seen in Figure 32, but any background. One way to accomplish this goal is through these four major steps:

1. The baseball should be extracted several times from current pitch frames so that there is a collection of baseballs in motion.
2. Many different background images should be collected.
3. Synthetic images should be created by pasting the extracted baseball onto the various background images in several different locations.
4. The CNN should then be retrained and possibly re-optimized.

Once these steps are completed, it is likely that the testing accuracy will go down or remain the same, as there are many more variables with the same proportion of data. If there is proportionally more data produced, there is a chance to even improve the current testing accuracy while also making the process dynamic and applicable in other environments.

5.2 Enhanced Baseball Position and Width Estimation

The method used to estimate the baseball's (x, y) pixel position and pixel width involved several image processing and filtering steps. These steps are computationally intensive as they are comprised of several filtering stages and several complicated and non-ideal techniques. A simpler method involves a sliding window approach. This method involves several less computationally intensive steps and could replace the object detection portion of this system if the current process cannot be made more efficient or if this new approach proves to be more accurate. The following steps would be applied to every frame that the CNN predicts a baseball was present.

1. Subtract the background (a frame grabbed shortly before the first pitch was classified to contain a baseball) from the current frame, as to keep only the baseball and noise in the image.
2. Convert the frame to black and white (i.e. all pixel values are either 1 or 0), while also making very low pixel values 0 to eliminate most of the noise.
3. Create a 30x30 pixel (approximately the minimum size of a baseball at the top of the strike zone) window.
4. Slide this window over the entire image with overlap and keep track of the max sum behind the window.
5. Move the window back to the spot with the max value, as it was assumed that this was the baseball.
6. Begin zooming out (i.e. increasing the dimensions of the window while still calculating a summation behind the window. For every increase in the window, ensure there was a minimal (empirically derived) increase in the sum of the frame behind the window.

Once the window was done growing, you have ideally found the baseball and its width. You could then use previously mentioned or possibly new techniques to remove returned objects that are not the baseball. The removal of outliers will be necessary as significant noise or an incorrect response from the CNN will result in not locating the baseball. Figure 39 on the left shows the located baseball before expanding the sliding window; in the middle, the full baseball was seen by zooming out, and on the right, the actual baseball in the corresponding frame can be visualized.

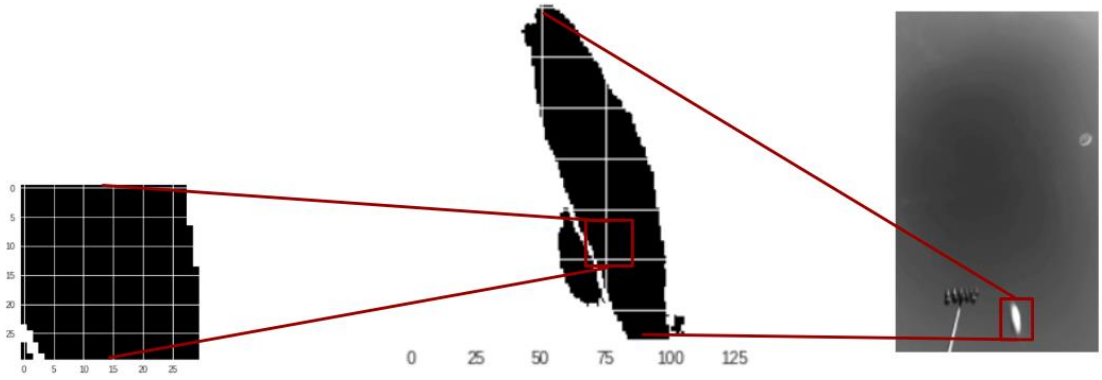

Figure 39: *Sliding window method for baseball detection.*

5.3 Real-Time Smartphone Application Implementation

Another future implementation for the smart home plate application could be to create a phone application that has the capability to run all the steps described in the methodology in real-time. The ideal smartphone for the task had to be determined. Table 14 shows the comparison between the Android operating system and the iPhone operating system (iOS). The decision between iOS and Android considered the app development process as if it were to be performed by a future team on WPI's campus. iOS was the clear winner, but only due to "Hardware Coding" requirement in the value proposition. This field refers to the ability to write code that would utilize the hardware in the phone. Based on extensive research, it was simple to develop using the slow-motion camera on an iPhone but requires bypassing the operating system when attempting the same thing on an Android device [29].

Table 14: *Value proposition for mobile phone operating system.*

Value Proposition	Out of 10	
Criteria	**Android**	**IOS**
Coding Learning Curve	5	5
Availability of Resources	6	4
Cost of Device	5	5
Camera Resolution	5	5
Camera Frame Rate	5	5
Hardware Coding	2	8
Total	**28**	**32**

6. Conclusion

The detection and parametric estimation of fast-moving objects required many steps from determining if the presence of an object exists to determining the object's location in 3D space. For the application with regards to a smart home plate in baseball, a methodology was constructed to go from a slow-motion recording of a pitch to an output of a strike or ball classification. By collecting the video, processing the frames, classifying the frames based on if they contained a baseball, determining the pixel location for the baseball, and using parametric estimation to convert these pixel locations to inch locations in 3D space, we classified a pitch as a strike or ball. The overall system was determined for the application of a smart home plate but can be generalized to apply to all fast-moving objects.

The system described in this report can be visualized below in Figure 40 as a final product concept diagram. The smartphone is resting on the home plate or possibly embedded into the home plate itself. Its FOV is encompassing at least the entire strike zone, and the pitch is being classified as a strike once the baseball intersects the strike zone volume.

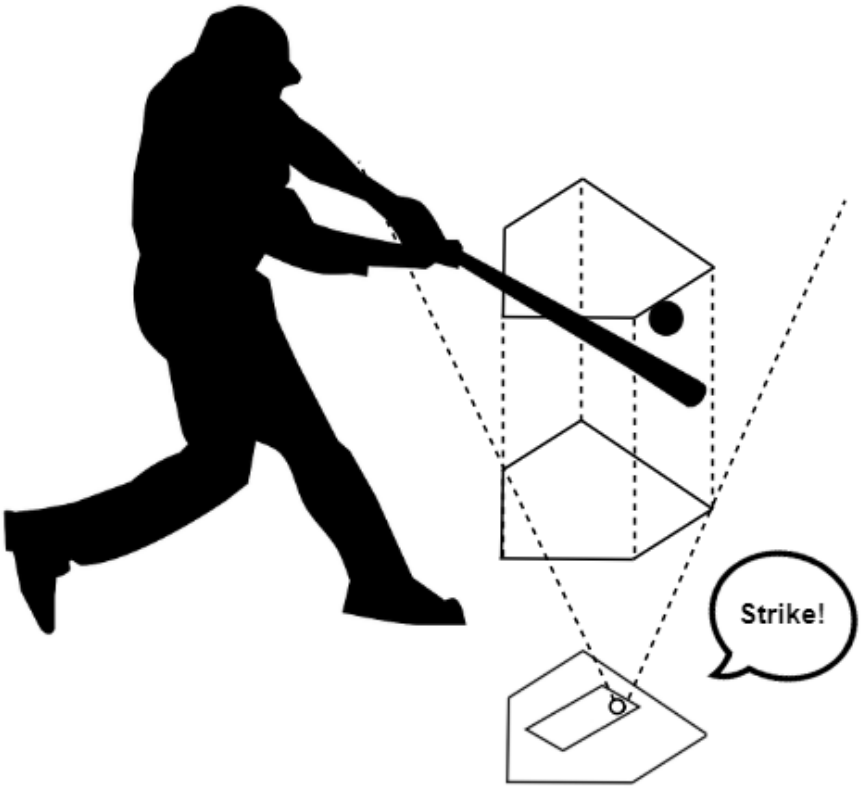

Figure 40: *The Smart Home Plate final concept diagram*

7. Appendix

Jupyter Notebook

This Jupyter Notebook contains all our system's code that takes in fisheye distorted images, classifies them as baseball present or not present, detects where the baseball is in each image, parametrically estimates where the baseball is located in 3D space, and then classifies the pitch as a ball or a strike. Link to notebook:

https://michaelpanicci.github.io/index.html

Please contact mapanicci@gmail.com if the link does not work due to further development.

8. Bibliography

[1] M. Chan. “Strike Zone.” wikipedia.com. https://en.wikipedia.org/wiki/Strike_zone (accessed Apr. 4, 2019).

[2] S. Alderson, C. Antonetti, S. Bernabe, J. Daniels, J. Dipoto, B. Gorman, J. Mozeliak, J. Schuerholz, J. Torre, M. Gaski, P. V. Mifsud Jr., T. Lepperd, “Official Baseball Rules,” MLB, 2019, USA, pp. 159. https://content.mlb.com/documents/2/2/4/305750224/2019_Official_Baseball_Rules_FINAL_.pdf (accessed Apr. 21, 2019).

[3] L. Berra, “Game Changers: An Electronic strike zone?”. MBL News, Nov, 30th, 2015. Retrieved from https://www.mlb.com/news/electronic-strike-zone-would-be-a-game-changer/c-158512610

[4] B. Eisenberg. “The Rapsodo Pitching Monitor for Pitcher Development.” TrueGrindSystems.com.http://www.truegrindsystems.com/pitcher-development-using-rapsodo-pitching-monitor-vs-diamond-kinetics-pitchtracker-vs-strike-smart-baseball/ (accessed Apr. 5, 2019).

[5] N. DiMeo. "Pitch f/x, the new technology that will change baseball analysis forever." Slate.com.http://www.slate.com/articles/sports/sports_nut/2007/08/baseballs_particle_accelerator.html (retrieved Oct. 18, 2012).

[6] M. Goder-Reiser and J. Prusaczyk. “Comparing the Rapsodo Baseball Device to Other Pitch Trackers.” Fangraphs.com. https://tht.fangraphs.com/comparing-the-rapsodo-baseball-device-to-other-pitch-trackers/ (accessed Apr. 18, 2019).

[7] “Strike - Above & Beyond.” jingletek.com. http://jingletek.com/official_website/html/en/ (accessed Mar 19, 2019).

[8] “Diamond Kinetics Pitchtracker.” maximumvelocitysports.com. https://maximumvelocitysports.com/products/diamond-kinetics-pitchtracker?variant=12964101816378&ut (accessed Apr. 22, 2019).

[9] A. Nathan. “Rapsodo, Trackman, and Pitch Tracking Technologies - Where We Stand.” Drivelinebaseball.com. https://www.drivelinebaseball.com/2016/11/rapsodo-trackman-pitch-tracking-technologies-stand/ (accessed Apr. 23, 2019).

[10] G. chifman. “The Lurking Error in Statcast Pitch Data.” fangraphs.com. https://tht.fangraphs.com/the-lurking-error-in-statcast-pitch-data/ (accessed Apr. 23, 2019).

[11] "Statcast: Glossary of terms". MLB.com. April 15, 2015. https://www.mlb.com/news/major-league-baseballs-statcast-glossary-of-terms-of-state-of-the-art-tracking-technology/c-118508858 (accessed Apr. 16, 2019).

[12] A. P. “Data deluge: MLB rolls out Statcast analytics on Tuesday.” USAToday.com. https://www.usatoday.com/story/sports/mlb/2015/04/20/Data-deluge-mlb-rolls-out-statcast-analytics-on-tuesday/26097841/ (accessed Apr. 18, 2019).

[13] “Statcast.” illinois.edu. http://baseball.physics.illinois.edu/statcast.html (accessed Apr. 22, 2019).

[14] D. Magdovitz. “FlightScope Premieres Advanced Tracking Technology for Baseball at ABCA Convention.” flightscope.com. https://flightscope.com/flightscope-premieres-advanced-tracking-technology-baseball-abca-convention/ (accessed Apr. 22, 2019).

[15] M. Roberti. “RFID Journal: Ask the Experts Forum - What is the accuracy of real-time location systems.” rfidjournal.com. https://www.rfidjournal.com/blogs/experts/entry?8102 (accessed Apr. 4, 2019).

[16] M. Roberti. “RFID Journal: Ask the Experts Forum - What can you learn from the NFL.” Rfidjournal.com.https://www.rfidjournal.com/articles/view?16586/&utm_medium%3Dnewsfeed_rss (accessed Apr. 4, 2019).

[17] “RFID Tag Sample Pack (UHF, Passive).” atlasRFIDstore.com. https://www.atlasrfidstore.com/rfid-tag-sample-pack-uhf-passive/? (accessed Mar. 27, 2019).

[18] “Soft Tag - RF - Pack of 500.” securitytags.com. https://www.securitytags.com/security-tags/fashion-security-tags (accessed Mar. 27, 2019).

[19] “Hall Effect Sensor.” electronics-tutorials.ws. https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html (Accessed Mar 19, 2019).

[20] R. S. Popović, “Hall plates and magnetoresistors,” in Hall Effect Devices, 2nd ed. Bristol, UK: IOP, 2004, ch. 4.

[21] “Lincoln 369737 Hall Effect Sensor.” WebstaurantStore.com. https://www.webstaurantstore.com/lincoln-369737-hall-effect-sensor/HP369737.html? (retrieved Mar. 27, 2019).

[22] “Honeywell Sensing and Productivity Solutions.” Digi-Key.com. https://www.digikey.com/ (retrieved Mar. 19, 2019).

[23] M. A. Zaveri, S. N. Merchant and U. B. Desai, "Multiple single pixel dim target detection in infrared image sequence," in *Proc. of ISCAS*, Bangkok, 2003, pp. II-II.

[24] S. A. Daud, N. H. Mahmood and P. L. Leow, "Wireless infrared proximity array sensor for three-dimensional surface reconstruction," *2014 IEEE 2nd International Symposium on Telecommunication Technologies (ISTT)*, Langkawi, 2014, pp. 144-148.

[25] “Excelitas Technologies Sensors LHI 968.” alliedelec.com. https://www.alliedelec.com/product/excelitas-technologies-sensors/lhi-968/70219618/? (retrieved Apr. 5, 2019).

[26] “HC-SR05/HY-SRF05 Precision Ultrasonic Sensor.” tindie.com. https://www.tindie.com/products/upgradeindustries/hc-sr05-hy-srf05-precision-ultrasonic-sensor/ (retrieved Apr. 12, 2019).

[27] “UT2F-EM-0A.” automationdirect.com. https://www.automationdirect.com/adc/shopping/catalog/sensors_-z-_encoders/ultrasonic_proximity_sensors/30mm_round,_6000mm_sensing_distance/ut2f-em-0a (retrieved Apr. 12, 2019).

[28] L. Segers, J. Tiete, A. Braeken, and A. Touhafi, “Ultrasonic multiple-access ranging system using spread spectrum and MEMs technology for indoor localization,” 2014.

[29] “A camera using the new camera2 API for Android Lollipop.” github.com. https://github.com/PkmX/lcamera (accessed Apr. 19, 2019).