USING PITCH TIPPING FOR BASEBALL PITCH PREDICTION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Brian Ishii

June 2021

COMMITTEE MEMBERSHIP

TITLE: Using Pitch Tipping for Baseball Pitch Prediction

AUTHOR: Brian Ishii

DATE SUBMITTED: June 2021

COMMITTEE CHAIR: Jonathan Ventura, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Paul Anderson, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Franz Kurfess, Ph.D.
Professor of Computer Science

ABSTRACT

Using Pitch Tipping for Baseball Pitch Prediction

Brian Ishii

Data Analytics and technology have changed baseball as we know it. From the increase in defensive shifts to teams using cameras in the outfield to steal signs, teams will try anything to win. One way to gain an edge in baseball is to figure out what pitches a pitcher will pitch. Pitch prediction is a popular task to try to accomplish with all the data that baseball provides. Most methods involve using situational data like the ball and strike count. In this paper, we try a different method of predicting pitch type by only looking at the pitcher's pose in the set position. We do this to find a pitcher's tell or "tip". In baseball, if a pitcher is tipping their pitches, they are doing something that gives away what they will pitch. This could be because the pitcher changes the grip on the ball only for some pitches or something as small as a different flex in their wrist. Professional baseball players will study pitchers before they pitch the ball to try to pick up on these tips. If a tip is found, the batters have a significant advantage over the pitcher. Our paper uses pose estimation and object detection to predict the pitch type based on the pitcher's set position before throwing the ball. Given a successful model, we can extract the important features or the potential tip from the data. Then, we can try to predict the pitches ourselves like a batter. We tested this method on three pitchers: Tyler Glasnow, Yu Darvish, and Stephen Strasburg. Our results demonstrate that when we predict pitch type at a 70% accuracy, we can reasonably extract useful features. However, finding a useful tip from these features still requires manual observation.

ACKNOWLEDGMENTS

Thanks to:

- Mom, Dad, Sister, and Family Dog Bailey
- All the teachers and mentors that have helped me over the years
- All the friends that remind me to enjoy life

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Like in all sports, baseball teams are trying to find any edge that can give them an advantage to winning. In baseball, teams are increasingly relying on statistics to make decisions in-game. This change was the subject of a 2011 film, Moneyball, a movie in which a team with a limited budget picked players primarily based on their statistics [7]. In more recent years, it has drastically changed the way baseball is played. For example, defensive shifts, or when a team stacks defenders on one side of the field, has increased in frequency from 12.1% in 2017 to 34.1% in 2020 [35].

1.1 Batting in Baseball

Batting in baseball is when a player faces a pitcher of the opposing team and tries to hit the ball the pitcher throws. However, it is easier said than done. As a batter, you only have a fraction of a second, sometimes 0.4 seconds, to figure out how fast is it going, where is it going, and should you try to hit it [33]. Doing all of this while the pitcher has different pitches that move different ways and at different speeds.

Batters often times try to predict what pitch is coming based on the current count (how many balls and strikes their are) or remembering previous at bats. This is the traditional way of trying to predict pitches. Another more difficult way batters try to predict pitches, is trying to catch the pitcher tipping their pitches.

1.2 Pitch Tipping In Baseball

Pitch Tipping is when "a pitcher unknowingly telegraphs information about the type of pitch he's about to deliver" [39]. The pitcher's "tell" or change in behavior usually involves something minute like how a glove is flared to the batter before the pitch or where the glove is when the pitcher comes to the set position before a pitch. The "tell" will continue to happen and be unrecognized until an observant team picks up on it and takes advantage of it. A pitcher is lucky if the pitching coach or teammate catch it first. However, most pitchers realize they are tipping their pitches when the batters of an opposing team seem to know every pitch that is coming. To fix a tell, the pitcher has to consciously train it out of their movements. Most pitchers are able to fix the issue; however, sometimes it comes back out of habit.

With teams studying film more than ever, recognizing a pitcher's tell is becoming more common. The most recent high-stakes example of pitch tipping was during a pivotal final game in the 2019 ALDS. This is where Tyler Glasnow tipped his pitches in a game against the Houston Astros. During the post game analysis, Alex Rodriguez, a former MVP baseball player, believed that the Astros "had something early and that was the difference in [the] game" [5]. Rodriguez meant that the Astros figured out how Glasnow was tipping his pitches, which helped them score and eventually win the game.

Another way of knowing what a pitcher is going to pitch is by stealing signs. To communicate information on what pitch the pitcher is pitching, the catcher will put a combinations of fingers down. Sign stealing is when the opposing team sees the catcher's combination and uses it to predict what pitch is going to be pitched. If a player or coach can see the signs while playing, it is fair game. It is even applauded when a player or coach use it to their advantage. However, using technology, like a

camera, to steal signs is illegal and frowned upon. One example of illegal sign stealing is the Houston Astros, the same team that found Glasnow's tip while he pitched [10]. The Houston Astros used a camera to read the signs from the catcher, and then they would signal the batter by banging a trash can [10]. After getting caught, the Astros management was suspended and the team was fined [10]. Many players felt the punishment was insufficient and were outraged [10]. It was not only that they cheated, but they were a dominant team during the time they were cheating, even winning a World Series. Though the Astros did cheat, they also legitimately found Glasnow's tip. It is important to note that using pitcher's tips is not illegal as long as the team is not using technology in real-time.

Though tipping pitches gives the batter a significant advantage, some batters may not want to know the tips. A Hall of Fame player of the 1950s and 1960s, Nellie Fox, "feared that being signaled would inspire him to muscle up ... and lead to a spate of warning-track flyballs" [39]. He did not want to mess with his mechanics and feared it would give him confidence and may lead to him swinging to hard and undercutting the ball when he hit. However, Fox is the exception, and most batters would want to know what pitch is coming.

Lastly, being able to recognize when a pitcher is tipping their pitches is useful for more than just the hitters. It is useful for a pitcher and the pitching coach to catch and fix tips before opposing teams find it. It is bound to happen to a pitcher, so it is better if their own team figures it out before the other teams have a chance to exploit it.

1.3 Contributions

Below is a list of the contributions from this paper.

- A method of collecting data for a pitch tipping dataset.
- Classification models for predicting pitch type based on a pitcher's set position
- An examination of the applicability of using pose data and object detection to discover and recognize a pitcher's tip.

Chapter 2

RELATED WORK

The related works covers five main related areas: find-grained activity recognition, traditional pitch prediction methods, pose estimation, object detection, and machine learning. Fine-grained activity recognition related works researched pitch prediction with full video clips of broadcasted baseball games. Traditional pitch prediction methods are discussed next. The pose estimation related work is OpenPose, which was used to estimate the pose of pitchers in the dataset. Using Detectron2, object detection was used to detect the pitcher and the pitcher's glove in the images. Lastly, for predicting pitches, we used discuss a machine learning model called the random forest.

2.1 Fine-Grained Activity Recognition

In one paper, the Piergiovanni and Ryoo studied fine-grained activity detection in recorded MLB games [27]. The activities included baseball activities like whether a pitch was as strike or ball. The paper created a dataset from 20 baseball games from the 2017 MLB post-season. With this dataset, they trained different models to classify video segments and segment continuous video.

In addition to the activity recognition, the paper also tried other experiments with the dataset including pitch type classification. They first tried using the I3D model, which is a CNN that uses video flow data as the input. Then, they used pose data of the pitcher's joint and body part locations as inputs to the InceptionV3 model.

Using the InceptionV3 model, they were able to get an accuracy of 28.4%. With the addition of sub-events as an input, the InceptionV3 model had an accuracy of 36.4%.

In a follow up paper [16], Chen et al. were able to improve the pitch prediction to around 57%. This paper build upon the first paper [27] by using a larger dataset and using the skeleton of the pitcher's pose instead of using heat maps of joints. One challenge that the paper looked at is oversampling in their baseball dataset. Fastballs are the primary pitch for most pitchers, and their dataset consisted of 55% fastballs. Since, fastballs are used more often then other pitches, it is important to account for potential biases in models because of the uneven distribution of pitch type classes.

Though these two papers experimented with pitch prediction, they used a different method than we are trying. They measured pitcher prediction for all pitchers. By using data from every pitcher, it is impossible to extract individual behaviors of one pitcher, which is what we want to accomplish. Additionally, the segments of video are not cropped to only the pitcher's initial movements, which means that movements after the pitch may have affected the model. Since tips are only useful when a batter can recognize them quickly enough, batters do not actively search for tips when the pitcher is about to pitch the ball.

2.2 Traditional Pitch Prediction Methods

Traditional pitch prediction involves using the current batter count (how many strikes and balls that batter has), among other things. We refer to this as traditional pitch prediction because this is what most people think of when predicting pitches. Additionally, this tends to be what a batter does while they are up to bat. Other things batters think about is who is the pitcher and what did they last throw? These types

of stats and more are recorded by trackers like PITCHf/x and the MLB's current tracker Statcast [26].

With the popularity and ease of use of machine learning models, there have been many different attempts at predicting pitches in baseball. From the few papers we looked at [32, 23, 31, 22], all of them used features in traditional methods to predict pitches. For example, in one paper, the features included batter count, what inning, how many outs, previous pitch type, previous at-bat result, pitch number and more [32]. These methods have been well researched, which is why we want to try a different method in using pitcher's tips.

2.3 Pose Estimation

Pose Estimation enables machines to visually detect humans poses in images. Pose estimation models look for key points on the human body like elbows in order to estimate the position in the image. One project that does pose estimation is called OpenPose [14].

2.3.1 OpenPose

OpenPose is an open source project for 2D pose detection for the body, foot, hand and facial keypoints of a person in an image [14]. OpenPose does this by using Part Affinity Fields (PAFs) and body part location estimations to learn to associate body parts with individuals in an image [15]. PAFs is "a set of 2D vector fields that encode the location and orientation of limbs over the image domain" [15]. First, OpenPose predicts confidence maps for body part detection and part affinity fields for parts association by inputting an image in their two-branch convolutional neural network

(CNN). Next, the body part predictions are connected via bipartite matching. For example, all predicted elbows and wrists will be a node in the graph with an edge from each elbow wrist. To ensure the correctly associated parts are connected, the orientation and locations from the PAFs are used.

The three main reasons for using OpenPose are that it is open-source, it has well-maintained documentation, and there is active development. OpenPose has detailed documentation, especially for the installation process. Most importantly, the active development helps us rely on their project more knowing that if there are bugs, they will be fixed.

We use OpenPose to detect the pitcher's pose in the videos we collected. Since it only looks for joints, it does not matter what jersey the player is wearing or any other differences in the video.

There are some potential downsides in pose estimation. The first drawback is that it is not perfectly accurate. Because it's an estimation, the error may make it hard to recognize small differences in the pitcher's delivery or make models less effective. However, the pose data is useful nevertheless.

2.4 Object Detection

Object Detection is a computer vision task that locates the presence of an object with a bounding box and determines which class or type of object is in the bounding box [12]. The models for object detection have improved rapidly and new techniques have been developed. In the past 5 years, there are two types of models that have had increasing success: R-CNN models and YOLO models.

The R-CNN or "Regions with CNN features" model generates region proposals (bounding-boxes) in the input image and inputs these regions into a CNN to predict what object is in the region [20]. This model worked well, but it was not very fast at training or predicting. To speed up the process, Fast R-CNN [19] and then Faster R-CNN [30] were developed. Fast R-CNN sped up the process by generating class predictions and bounding boxes for the proposed region at the same time [19]. However, Fast R-CNN still needed the region proposals as input. Faster R-CNN combined the region proposal network with the class prediction network, so both could work on the same output of a CNN [30].

The YOLO or "You Only Look Once" model is a "single neural network [that] predicts bounding boxes and class probabilities directly from full images in one evaluation" [28]. Though this may be a lower accuracy, the single neural network allows for fast training and evaluation. The YOLO model can process "images in real-time at 45 frames per second" and a smaller version, Fast YOLO, can process "155 frames per second while still achieveing double the mAP [mean average precision] of other real-time detectors" [28].

For this project we chose to use Detectron2 for object detection because it's built upon the Faster R-CNN architecture. Since we do not need it to run in real-time, there is not a reason to use a YOLO model. Additionally, Detectron2 has a wide range of features and is well maintained and supported.

2.4.1 Detectron2

Detectron2 is Facebook AI Research's object detection and segmentation research library [41]. Detectron2 improved on two previous projects: Detectron [21] and Mask R-CNN (an extension of Faster R-CNN) [24]. Detectron, released in 2018,

was Facebook's object detection library that was built using the Caffe2 deep learning framework [21]. Mask R-CNN was a library that helped create detection and segmentation models using PyTorch 1.0 that trained twice as fast as the original Detectron [24]. Detectron2 followed Mask R-CNN in being implemented in PyTorch as well. In addition, it was designed to have a modular design that allows for many different applications like instance segmentation, semantic segmentation, and even human pose prediction [42]. The researchers of Detectron2 focused on making is very user friendly and easy to use. This included a `Dockerfile` with a corresponding `README.md` that made setup quick.

The primary use of Detectron2 for this project is to detect the pitcher's baseball glove position throughout the pitch. One of the most common areas batters look at for pitch tipping is the glove. We will primarily be looking at the position of the glove relative to the OpenPose position. This means we will save the glove's position and mask. Additionally, we will be tracking and recording the pitcher's position and mask.

2.5 Machine Learning

Machine learning is a very diverse field with many different models and algorithms to use. For the purposes of this paper, we will be using a supervised machine learning classifier. We are using a classifier because predicting which type of pitch is thrown is a problem where the output belongs to one of the two classes (did they throw a fastball or a curveball?) [11]. Since some pitchers pitch more than two pitches, we will need to use models that can handle multi-class classification or problems with more than two classes [11]. The reason we are using a supervised machine learning model is because we have "prior knowledge of what the output values for our samples

should be" [34]. In other words, we know the pitch type that is pitched, so we are trying to map the inputs to the correct output.

In this paper we will be using the Random Forest machine learning model. Since we are trying to predict which type of pitch is being thrown, the model will be a classifier.

2.5.1 Random Forest Classifier

Random forest is an algorithm that uses ensemble learning. Ensemble learning is using multiple machine learning models instead of one to make predictions [13]. In the case of the random forest algorithm, it is an ensemble of decision trees (hence the forest name).

Random forest creates each decision tree using a bootstrap sample (sampled with replacement) from the training set [13]. When creating the decisions in the tree, random forest chooses a random subset of input features to base the decision on [13]. This ensures that each of the decision trees will be random and be less correlated to one another.

Chapter 3

DATASET

3.1 MLB Film Room

The MLB Film Room is the new video archive for MLB baseball that was introduced to the public on September 8, 2020 [9]. It has over 3.5 million videos, which includes video from every pitch thrown in MLB since 2017 [8]. The video clips can be searched using many different types of filters like pitcher, pitch type, team, date, and more. For example, we can use the following filters to get every fastball by Tyler Glassnow against the Houston Astros in 2019 using the following filters: 2019 season, batting team name as Houston Astros, pitching team name as Tampa Bay Rays, player as Tyler Glasnow, and pitch type as fastball.

The fine-grained activity recognition papers [27, 16] used YouTube videos of the MLB games and segmented the videos manually or trained a model to segment the videos. Like those papers, the MLB Film Room segments each pitch into individual videos. The reliability and guarantee that the segmented video is correct is an added benefit of the MLB Film Room. The most important feature is the filters. Using filters, we are able to label the pitches by pitch type and to get specific games. The only challenge was trying to download the videos. With a little investigating with web browser developer tools, we figured out that the videos were hosted on Streamable.com. Streamable.com allows users to download and crop the videos when the video has a Streamable.com link. Using some JavaScript scripts, we were able to parse all the links to crop and download the videos. The videos were 60 frames per second at a resolution of 1280 by 720 (720p). Though this process is still manual,

there was no direct way to download the videos from MLB Film Room and this way was somewhat efficient.

The one downside of the videos from MLB Film Room is that it is video from the live broadcast. This means that the pitcher's whole windup is not always in the video. In addition, the broadcast view is behind the pitcher instead of from the batter's perspective. This means that there are things that the batter can see that the video cannot and vice versa. However, the video is still adequate for finding certain pitcher tips.

3.2 Pitchers

The selection of pitchers are important. To train the models, we wanted a consistent pitching routine and a pitcher that throws a lot of pitches. The pitchers that pitch the most pitches are starting pitchers. However, most starting pitchers have a different movement when pitching with no one on base (pitching from the wind-up) and pitching with a runner on base, which is known as pitching from the stretch. There are some pitchers that only pitch from the stretch regardless of whether a runner is on base. We picked three pitchers that pitch exclusively out of the stretch: Tyler Glasnow, Yu Darvish, and Stephen Strasburg.

3.2.1 Tyler Glasnow

Tyler Glasnow is a right handed pitcher for the Tampa Bay Rays. We chose Tyler Glasnow because he pitched in the 2019 playoffs where he unknowingly was tipping his pitches. He was also the inspiration for this project. Tyler Glasnow throws three

Table 3.1: Glasnow Pitch Percentage 2019 [3]

Pitch	Pitch %
Fastball	67.2%
Curveball	29.3%
Changeup	3.5%

different types of pitches: fastball, curveball, and change up. However from Figure 3.1, we can see in 2019 he mainly pitched his fastball and curveball.

For the Glasnow dataset, we chose his games during the 2019 ALDS against the Houston Astros. During game 1, he did not perform well giving up 4 earned runs in 4.1 innings for an ERA of 4.15 [6]. He did even worse during game 5 where he gave up 4 runs in 2.2 for an ERA of 13.50 [6]. It was in this game 5 where there was public speculation of Glasnow tipping pitches [5]. It was after this game where Alex Rodriguez said, "to me, the Astros had something early and that was the difference in the game" [5]. From this dataset, we expect that we will be able to predict pitches somewhat accurately because there is a confirmed tip.

3.2.2 Yu Darvish

Yu Darvish is a right handed pitcher in the MLB. Currently, he is pitching for the San Diego Padres. He has a wide variety of pitches, but mainly throws 4 types of pitches: 4-seam fastball, slider, 2-seam fastball (sinker), and cutter.

Table 3.2: Darvish Pitch Percentage 2017 [2]

Pitch	**Pitch %**
4-Seam Fastball	35.5%
Slider	25.0%
2-Seam Fastball	16.6%
Cutter	14.4%
Curveball	5.7%
Changeup	2.6%
Eephus	0.2%

Though he is currently pitching for the Padres, we are going to investigate his pitches during his 2017 season with the Los Angeles Dodgers, specifically during the World Series. Darvish struggled during the World Series, giving up a total of 8 earned runs in 3.1 innings for an ERA of 21.60 [38]. This is significantly worse compared to his ERA of 3.86 for the entire 2017 season. This may have been the result of Darvish pitching poorly, but there is another possible reason for his struggles. The team he was facing in the World Series was none other than the Houston Astros. 2017 is one of the years that the Houston Astros were confirmed to be cheating [38]. After the news of the Astros cheating, Darvish was quoted saying, "A couple of Astros players told me that I was tipping pitches 100%. But then it came out they're stealing signs. That's why I want to know, (if) the World Series came from stealing signs or tipping pitches." [38].

Our goal is to be able to help Darvish come to some sort of conclusion to whether he was tipping pitches in a similar style to Glasnow or if the Astros were stealing signs (there still is a chance he is tipping his pitches in another fashion). For his games, we chose the two playoff games Darvish pitched against the Houston Astros and his last game in the regular season against the San Diego Padres.

3.2.3 Stephen Strasburg

Stephen Strasburg is a pitcher for the Washington Nationals. Strasburg is a four pitch guy that pitches them all somewhat consistently as seen in Table 3.3.

Table 3.3: Strasburg Pitch Percentage 2019 [4]

Pitch	Pitch %
Curveball	30.7%
4-Seam Fastball	28.6%
Changeup	20.7%
2-Seam Fastball	19.7%

Strasburg does occasionally tip his pitches before coming to the set position. However, his run in against the Houston Astros fared better than Glasnow and Darvish. In the game, Strasburg was tipping his pitches in a way that was well known to opposing teams and his own pitching coaches. Thankfully for Strasburg, National's coordinator of advance scouting, Jonathan Tosches also saw the tip [40]. After a quick adjustment, Strasburg did not give up another run. Unfortunately, this tip is not seen in broadcast video because it's before Strasburg comes set. However, we want to see if he has any tells when he is set. We chose the two games before facing the Houston Astros that Strasburg pitched against the Dodgers. These were both home games for the Dodgers, so the camera angle would be fairly consistent between the two games.

3.3 Feature Extraction

In order to extract usable features from the video clips, we used Detectron2 and OpenPose. Detectron2 is used to detect the pitcher and the pitcher's glove in each frame. After, we used OpenPose to estimate the pose of the pitcher in each frame. In sections 3.3.1 and 3.3.2 we discuss the usages of Detectron2 and OpenPose, respectively.

3.3.1 Object Detection Data

(a) (b)

Figure 3.1: Example of a frame without (a) and with (b) object detection.

We used the `DefaultPredictor` in the Detectron2 Library to detect where the pitcher and the pitcher's glove was located in the image. The `DefaultPredictor` uses a configuration object that has a model and other parameters. The configuration we used is Detectron's COCO-InstanceSegmentation's `mask_rcnn_R_50_FPN_3x` model. This model is a Mask R-CNN with a Feature Pyramid Network (FPN) for feature extraction. It was trained on top of the ResNet-50 model for approximately 37 epochs [41].

When running Detectron2, it finds multiple objects. In order to differentiate between the objects we want and do not want, we only saved the objects that matched two criteria: all points of the object's box must be between pixel 200-750 horizontally and 200-700 vertically and the object detected must be classified as 0 (person) or 35 (baseball glove). This was a simple solution that worked well for the limited scope. However, we would recommend training a model or having additional filters to catch mistakes that may occur in a larger scope or a more general application to all pitchers.

In Figure 3.2, there are boxes and masks for the pitcher and the pitcher's glove. Blue represents the pitcher's mask and box. Red represents the pitcher's glove's mask and

box. From Detectron2, we kept the box and mask of the pitcher and and the pitcher's glove. When storing the masks, we recommend saving them as images and not as `numpy` arrays because the `numpy` arrays are significantly larger.

3.3.2 Pose Data

(a) (b)

Figure 3.2: Example of a frame without (a) and with (b) pose data.

Using OpenPose we are able to detect the pitcher's body position in each frame of video. We ran OpenPose with tracking on each frame to improve visual smoothness. This does not have an effect on the efficiency. Additionally, we set the number of people in the frame to 1. Since we already have the position of the pitcher from the Detectron2 data, we mask each frame with the box of the pitcher. This ensures that we will only have person to track and makes the skeleton data consistent. The data we save includes both the json output of each frame as well as a video with the pose for each frame. The JSON output is what is used in analyzing the pitcher's movements, and the pose video is used for debugging purposes.

There are 25 of keypoints that are collected each frame. Each keypoint has an `x` position, `y` position, and a confidence value assigned to it.

Chapter 4

IMPLEMENTATION

For the implementation, we tested four different frames: two for Glasnow, one for Darvish, and one for Strasburg. All three pitchers have a model for their set position. The fourth model is when Glasnow is in the pitching motion in his leg kick. While looking at his pitches, we thought when he reaches the peak of his leg kick, he gives away what pitch he is pitching. We wanted to test our hypothesis. For the rest of the implementation sections, we will discuss libraries that were used, the data preprocessing steps, and the machine learning models used.

4.1 Libraries

In addition to Detectron2 and OpenPose, there were many other libraries used for the implementation. The main libraries used includes, OpenCV, pandas, and scikit-learn. The scripts and Juptyer Notebooks were written in Python 3.9.1.

4.1.1 OpenCV

OpenCV is an "open source computer vision and machine learning software library" that is used by more than 47 thousand users including companies like Google, Microsoft, and Toyota [37]. OpenCV was initially released by Intel in early 2000s. OpenCV is mainly used to get frames from a video, add masks to images, crop images, and any other video and image related solutions.

4.1.2 pandas

pandas is a "fast, powerful, and easy to use open source data analysis and manipulation tool" [25]. The main feature that we used is its `DataFrame` object, which is like a excel spreadsheet for coding. These were used to handle any data manipulation, cleaning, and organizing for the models.

4.1.3 Scikit-learn

Scikit-learn is a library for machine learning in python. It includes all the tools that one would need including supervised learning models for classification and preprocessing tools for normalization and feature extraction [18]. We used this library for the Random Forest model, for splitting our data, for visualizing our data, and for feature importance out of the model.

4.2 Preprocessing Data

Before training models and testing them, we had to preprocess all the data. This is an important step to remove extraneous data and to make sure the data is consistent. We did this in three steps: Finding the set position, normalizing the data, and cleaning the data. Each step is discussed in detail in the next subsections.

4.2.1 Finding The Set Position

All the data that has been collected is from videos with different sizes. This means that pitcher's motion is different for each video. In order to get frames from the same part of the pitcher's motion, we needed a method to figure out a common pose the

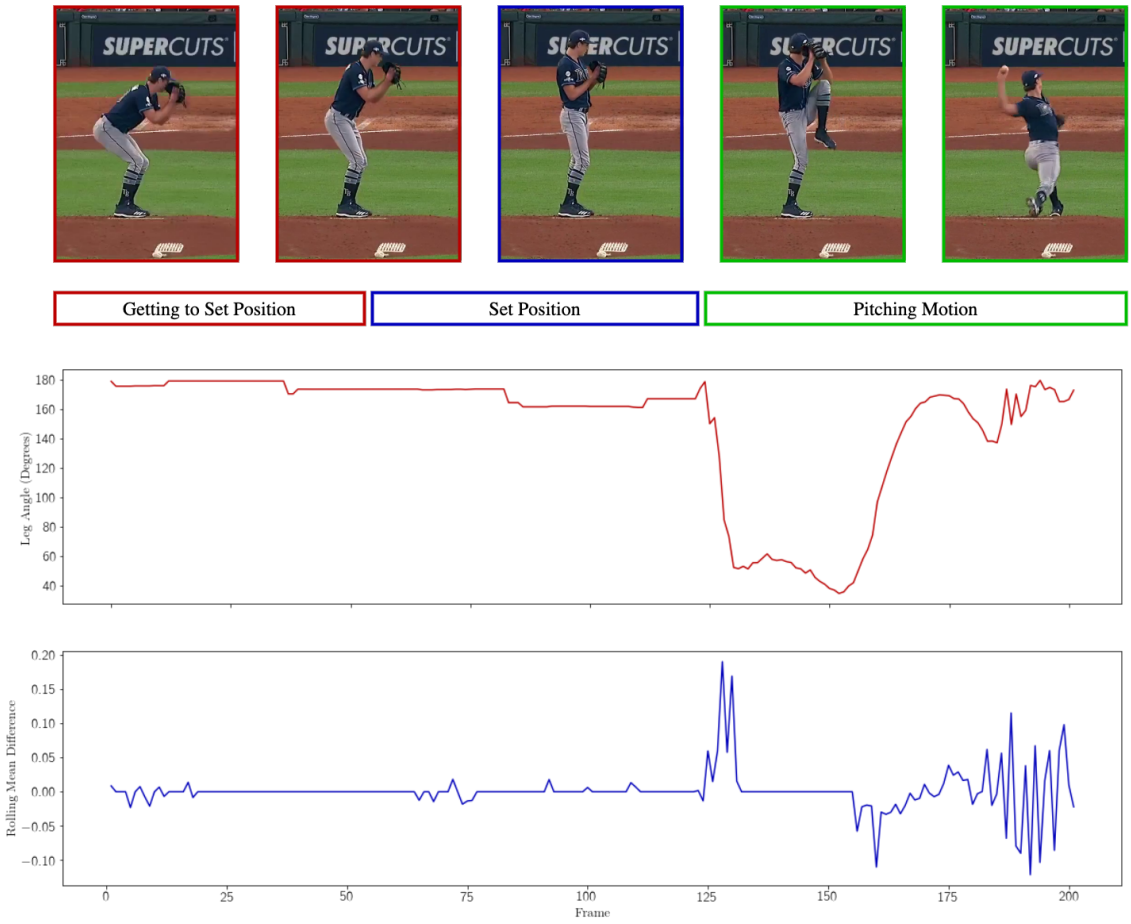


Figure 4.1: The three different states of the pitcher. Below is a graph of the leg angle (red line) and rolling mean difference from the left knee's x position (blue line) throughout each state.

pitcher was in at a certain point in each video. After we get the common pose, we can locate the set position, which is what we will analyze later.

First, we defined a few distinct states that the pitcher can be in: getting to set position, in set position, and the pitching motion. The three states can be see in Figure 4.1, where getting to set position is in red, set position is in blue, and the pitching motion is in green. Because broadcast video is not a constant feed of pitcher, the getting to set position state and the majority of the in set position state may not be captured in the video. However, the pitching motion is almost always in the broadcast video. Since the pitching motion is reliably in all the videos, we created

a method of locating the pitching motion. The pitching motion of a pitcher will always include a leg kick of some kind. A right-handed pitcher will have a left leg kick and a left-handed pitcher will have a right leg kick. With the pose data, we are able to see the pitcher's leg kick from the video clip in the JSON data. We used a pandas `Dataframe` to merge all the JSON data from OpenPose to do this. First, we calculated the leg angle of the kicking leg of the pitcher. Then, we located the frame with the smallest leg angle. This did not always work, so we added a check to make sure the heel was below the wrists of the pitcher to catch an edge case when the pitcher has thrown the ball. In Figure 4.1, the leg angle is the graph with the red line, which shows how the leg angle changes throughout a video clip. The minimum leg angle in this video clip is frame 152, which is the fourth image from the left in Figure 4.1.

After we know where the leg kick has occurred, we look at the kicking leg knee to see when it first moved horizontally. We look for this because this movement is the start of the pitching motion. In the MLB rule book, it states that "after assuming Set Position, any natural motion associated with his delivery of the ball to the batter commits him to the pitch without alteration or interruption" [17]. This means that once the pitcher moves the leg, the are committed to pitching the ball. The pitcher can also try to pick off a runner on base, but those video clips were excluded in the dataset. To find when the leg starts moving, we calculate the difference rolling mean of the 5 frames before and after each frame. Since, the pitcher does not move their kicking leg until they start the pitching motion (if they move it and stop it is considered a balk [17]), we look for the frame that has a difference rolling mean of 0 before the frame and a difference rolling mean greater than 0 after the frame. This frame is the last frame where the pitcher is set, which is what we used as our frame to analyze. From Figure 4.1, we can look for the first frame that meets this condition in the graph with the blue line. Starting from frame 152 and going to the left, we

can see around frame 125 is the first location where the frames before have a rolling mean of the difference of 0 and the frames after have a rolling mean of the difference of greater than 0. The actual frame for this video is frame 124.

This method of getting the last set position frame worked well. Though we would have liked to analyze more of the set position, the broadcast video did not provide enough videos of the pitcher getting to set position. Having this information would have allowed us to analyze the pitcher during the entire duration of the set position.

4.2.2 Normalizing Data

The next step of preprocessing is normalizing the data. It is necessary to normalize the poses of the pitcher because the pitcher is in different locations in each video. This may be because the pitcher is pitching at a different stadium or because the broadcast is zooming in or out on the pitcher. To adjust for the differences, we normalize the data.

We normalized the data by taking the smallest and largest value in both the `x` and `y` direction. The smallest value became 0 and the largest value became 1. The `x` and `y` values in the data were normalized between these two values. For example, if y_0 is the smallest `y` value and y_1 is the largest `y` value, then the value of y_i will be $(y_i - y_0)/(y_1 - y_0)$.

4.2.3 Cleaning Data

The last step in preprocessing the data is cleaning it. This means a few things including removing low confidence values from the datasets and checking for `NaN` values.

The reason we want to remove low confidence values is because we want to make sure we are using the data that has the best estimations. If the values have a low confidence, they may be unreliable, and thus not provide useful features for the models to use. There were four keypoints that did not have a confidence value over 50%: LEar, LEye, REar, and REye. These keypoints (their `x` and `y` values) were removed from the features that the models used.

Next, we had to make sure there were no `NaN` values in the dataset. The pose data did not have any `NaN` values, however, the glove data did. Unfortunately, Detectron2 did not always capture the pitcher's glove. To fill these `NaN` values, manual user input was used. We created a simple script that would take the set position frame and allow the user to click where they thought the glove was located in the frame. This position was used for the `NaN` values.

Another issue we had with the glove data is that Detectron2 would not always capture the whole glove. In many cases, the glove was only partially masked and the bounding box only included half the glove. This limited us to only using the glove's top `y` position.

Lastly, there was a check of the actual frame and if it looked like the set position. There were some videos that do not have the set position in the video, and thus finding the set position step picks the closest frame to the set position. Each of the frames were manually inspected, and any frame that had the pitcher in the pitching motion was removed from the dataset.

Table 4.1: Final Pitch Distribution

Pitch	Glasnow	Glasnow-leg	Darvish	Strasburg
4-Seam Fastball	49	74	29	25
2-Seam Fastball	N/A	N/A	30	30
Curveball	20	31	3	50
Changeup	N/A	N/A	2	33
Slider	N/A	N/A	43	N/A
Cutter	N/A	N/A	31	N/A
Total	**69**	**105**	**138**	**138**

4.3 Random Forest Models

There were four random forest models that were created for each pitcher. The first model's features included both the pose data and glove position (y-position of the top of the glove). The second model used only the pose data. The third model used only the elbow and wrist data. The last model used only the glove data. For the random forest models, we used 50 estimators (number of decision trees in the model). For `max_features`, or the number of features to use to use when splitting in the decision tree, we used the square root of the number of the features in the dataset.

4.4 Splitting Data

When training models, it is important to split your data in a way where the model can be tested with data it has not seen. We used two methods to do this: train-test split and stratified k-fold split.

4.4.1 Train-Test Split

For Glasnow's dataset, we chose a train test split for the model. Since we know that he was tipping his pitches, we used the previous game as the training set and used the game he tipped his pitches as the test set.

4.4.2 Stratified K-Fold Split

Stratified k-fold split is a variation of cross validation that is an alternative method to train-test split to test your model. Cross validation separates your dataset into different folds and splits the data into a train and test set. It runs your model with each fold as the test set and all the other folds as the training set in an iterative process. Stratified split creates different folds than just splitting the data into separate folds. Stratified splits ensure that each fold gets approximately the same percentage of samples for each pitch type. This ensures that no fold is all one pitch type. The K-fold part of the split allows us to split the dataset into k number of folds. We used a 5 fold split. We used stratified k-fold split for the Darvish and Strasburg datasets because there was not a specific set of pitches that we wanted to test. Instead, we wanted to test the data for any tips. This allowed us to test each fold of the dataset by using the other folds to train the model.

Chapter 5

RESULTS

For the results, we used accuracy to measure the performance of the model for the test set. Additionally, for the binary classification models, we used Matthew's correlation coefficient (MCC) as an additional metric. We chose to use MCC as an additional metric because accuracy is not perfect. For example, if 70% of pitches are a fastball, the model can guess all fastballs and be 70% correct. MCC takes into account the entire confusion matrix of true and false positives and negatives. A value of +1 is perfect prediction, 0 is as good as guessing, and -1 is getting every prediction wrong.

5.1 Glasnow Results

Glasnow's results are split into two different sections: Glasnow in the set position and Glasnow during the pitching motion. To see a visual representation of the features that were used for the set and pitching motion models see Figure A.1 and Figure A.2, respectively. Most speculation of Glassnow's tip is expected to be the glove height and angle [5]. From this, we expect the Only Arms and Only Glove models to perform the best.

5.1.1 Glasnow Set Position

Table 5.1: Glasnow Set Results

Features	Test Accuracy %	MCC
Only Pose Data	59%	-0.09
Only Arms	77%	0.31
Only Glove	64%	-0.04
All Data	73%	0.10

In Table 5.1, we can see that the model using only the arms performed the best at a 77% accuracy. There were two surprising results. The first surprise is that the glove only model did not perform similarly to the arms model. Since the tip is supposed to be based off the position of the glove, we expected this model to perform well (and it's connected to one of the arms). This means that either our glove positions were inaccurate or that the tip is more an the beginning of the set position and not towards the end of the set position, which both could be true. The second surprising result is the all data did not perform the best or equal to the best. Since the all data model does have the arms in its set of features, it should be able to perform similarly. One explanation could be that we were not using enough trees or that we had too many trees such that the arms did not have as much as a significance (which helps prevent bias). Overall, these results are promising. These results were not only successful, they validate the proof of concept.

5.1.2 Glasnow Leg Position

Table 5.2: Glasnow Set Results

Features	Test Accuracy %	MCC
Only Pose Data	72%	0.35
Only Arms	58%	0.04
Only Glove	58%	0.00
All Data	69%	0.30

In Table 5.2, we can the model using only pose data performed the best with an accuracy of 72%. This is another great result. All data also performed similarly at 69%. We hypothesis that the all data did not perform as well because we are only using 50 trees in our random forest. Add glove data, only adds one new feature, but that was enough to decrease the accuracy by a few percent. What is interesting to see is that the whole pose was more telling than the just the arms like in the set position. After running feature importance, the four most important features were Nose_y, RElbow_y, RShoulder_y, and RWrist_y. The nose position is interesting, but the other three make sense. When Glasnow is doing his leg kick, if his right hand is higher, he is throwing a fastball. If it is lower, he most likely is throwing a curveball. It is noticeable in the broadcast image because the glove covers his ear for most of his fastballs, and his ear is visible when he is throwing his curveball.

5.2 Darvish

Darvish's results are split into two different sections: Darvish in the set position with multi-pitch classification and Darvish in the set position with binary pitch classification. We tested binary classification so we can compare the results with Glasnow's

results. A visual representation of the features for Darvish in the set position can be seen in Figure A.3.

5.2.1 Darvish Multi-Pitch Classification

Table 5.3: Darvish Multi-Pitch Set Results

Features	Test Accuracy %
Only Pose Data	36%
Only Arms	28%
Only Glove	32%
All Data	37%

In Table 5.3, we can see that all the models performed around 30-35% accuracy. Both the only pose model and all data model performed the best, which is expected because it has the most features. We were unsure whether Darvish was tipping his pitches, and with these results, we can say that Darvish was not tipping his pitches in a similar way to Glasnow. We came to this conclusion because the accuracy of these models were significantly worse than Glasnows. However, the models did perform better than guessing.

5.2.2 Darvish Binary Pitch Classification

For binary pitch classification, we split the pitch type between fastball and off-speed pitches. 4-seam and 2-seam fastballs were classified as fastballs and all the other pitches were classified as off-speed pitches. We chose to run this test to get a more similar comparison between Darvish and Glasnow. Because Darvish throws more pitches than Glasnow, the accuracy will be lower because there are more choices to choose.

Table 5.4: Darvish Binary Pitch Set Results

Features	Test Accuracy %	MCC
Only Pose Data	56%	-0.07
Only Arms	56%	-0.07
Only Glove	55%	0.09
All Data	49%	0.09

In Table 5.4, we can see the binary classification results for Darvish. From the results, we can see that the accuracy of all the models are around 50% and are similar to guessing.

5.3 Strasburg

Similarly to Darvish's results, Strasburg has a section for multi-pitch classification and binary classification in the set position. A visual representation of the features for Strasburg in the set position be seen in Figure A.4.

5.3.1 Strasburg Multi-Pitch Classification

Table 5.5: Strasburg Multi-Pitch Set Results

Features	Test Accuracy %
Only Pose Data	41%
Only Arms	40%
Only Glove	44%
All Data	51%

In Table 5.5, we can see that models performed better than Darvish's results. Because Strasburg pitches four different types of pitches these are pretty surprising results. This means that the model is almost twice as good as guessing.

5.3.2 Strasburg Binary Pitch Classification

For binary pitch classification, we split the pitch type between fastball and off-speed pitches. 4-seam and 2-seam fastballs were classified as fastballs and all the other pitches were classified as off-speed pitches.

Table 5.6: Strasburg Binary Pitch Set Results

Features	**Test Accuracy %**	**MCC**
Only Pose Data	64%	0.36
Only Arms	61%	0.27
Only Glove	67%	0.19
All Data	69%	0.32

In Table 5.6, we can see that Strasburg's results were somewhat comparable to Glasnow's results. This is interesting because we knew there was a tip from Strasburg as he comes to the set position, but we were not sure if the tip still showed in the set position. At least for throwing fastballs, there may be some remnants of the tip. After using a feature extraction tool on the random forest models, we found that glove_y was the most important feature in the dataset. This was a surprising result, and after manually investigating the frames, we found a slight difference in the glove height between fastballs and off-speed pitches as seen in Figure 5.1. In Figure 5.1, we can see the glove is higher for the 2-seam fastball than for the curveball (note that not all pitches are this drastically different). Though the model does not differentiate between the specific pitches in each of the categories, it is impressive results that were not expected.

(2-Seam Fastball) (Curveball)

Figure 5.1: Strasburg's glove height difference between pitches

Chapter 6

DISCUSSION

6.1 Limitations

Though we were able to get promising results, there are a few limitations that we also need to include. The first limitation is the size of our dataset. The main reason for the small dataset is that the manual step is very time consuming. To make the feature extraction steps run smoothly, each clip was cropped to remove irrelevant frames of the broadcast that were in the video clips on the MLB Film Room. Roughly a quarter of the video clips of pitches were thrown out completely because the set position was not seen or it was an atypical angle of the pitcher. This also highlights the other limitation of the duration of the video clips, particularly while the pitcher is in the set position. Most usable tips are found in the set position because the batter has enough time to see it and make a decision before the pitch is thrown. However, we were only able to analyze the last frame of the set position because that was the most consistent part we could get.

The last limitation we wanted to discuss was the view of the pitcher. The broadcast view is from behind the pitcher and not the view the batter has. This limits us to focus on things that both the batter and the viewer from the television would be able to catch, like the height of a glove. It would have been interesting to get a closer perspective to a batter.

6.2 Threats to Validity

We identified two main threats to the validity of the models: potential biased data and estimation error.

When using a dataset, there is always a chance for biased data. With baseball pitch types, the majority of pitches are fastballs, which already adds some bias. A common problem to have is having a model only predict fastball. This may give the model a decent accuracy, but low precision. To make sure our models did not just guess fastball, we monitored the F1 scores and confusion matrices. Additionally, we used stratified splits for Darvish and Strasburg's models.

The other threat is the pose and glove estimation error. Though we believe the pose and glove positions were fairly accurate, there is a change that there is some error. Since tipping a pitch may be a subtle difference, this error may have improved or worsened the model's effectiveness. To mitigate the threat, we threw out values that had a low confidence like the eye positions, ear positions, and the glove `x` values.

Lastly, there are some other minor threats to validity like the different camera angle at different stadiums. We tried to keep this to a minimum, and only Darvish has one game at a different stadium. Both Glasnow and Strasburg's games were at the same stadium.

6.3 Ethical Considerations

As technology is being used more and more in sports, we must make sure to always consider ethical issues that may arise to use of the technology. Especially after the

Houston Astros blatantly cheating with cameras, it is important to reiterate that using technology to steal signs is very illegal and should not be done.

However, our approach in using tipping pitches is not illegal and is applauded in the baseball community. As long as you do it with your own eyes in real-time during a game, there is nothing wrong in doing it. Even using pose estimation on video after or before a game is allowed. Though there is not public information about teams using machine learning to find pitchers tipping their pitches, we believe a team would be handicapping their players if they did not try or have a method already in place. Outside of the game, as much research is allowed. Just don't use your model in real-time and bang a trash can.

6.4 Convolutional Neural Network

The last topic we would like to discuss is convolutional neural networks. We did try using a CNN to predict the pitches, however we were not successful in getting it to work (it was predicting fastballs every time). We believe this problem is mainly because of the small dataset. Maybe if we had around 1000 pitches, it would have performed correctly. We did not pursue this because of time and because we were unsure how useful it would actually be. The challenge with the CNN is how do we extract useful features that we can give to players to use. By using pose and glove data, we are able to extract meaningful (to humans) data that we can see. A tip has no value if we as humans can not recognize it. Ultimately, that is why we did not pursue using a CNN any further.

Chapter 7

CONCLUSION

This study tried a new method of predicting pitch type by using the pitcher's pose and glove position in the set position. We tested our method on three pitchers: Tyler Glasnow, Yu Darvish, and Stephen Strasburg. Tyler Glasnow's tip was expected to be caught in the model, however, we were unsure about Strasburg and Darvish's tips. To make predictions, we used a random forest classifier to predict the pitch type based on pose and glove height data. In the results, Glasnow's models had better accuracy than Darvish and Strasburg's models. This is most likely because of the tip. However, another reason for the higher accuracy is that Glasnow only throws two types of pitches as opposed to four or more. To have a more similar comparison, we converted Darvish and Strasburg's pitches to a binary classification problem. We did this by converting both types of fastballs to one class and all other pitches to the other class. Darvish's results were similar to guessing, however Strasburg's results were similar to Glasnow's results. From feature importance tools, we discovered that Strasburg has a tendency to have his glove higher when he pitches fastballs, which was a new tip we did not expect. Overall, these are promising results that were discovered, and we see believe there are further improvements that can be made to better identify a pitcher's tip.

7.1 Future Work

This work showed that there is validity in predicting pitches based on the pitcher's pose and glove position. However, there are many potential improvements to make it useful to batters and pitchers alike.

7.1.1 Dataset

The first improvement would be a dataset that shows the batter's perspective or another front facing view of the pitcher, like from the player's dugout. This would provide the models with data that a batter sees, and thus any useful features extracted would be easier to apply in practice. In addition, we would be able to analyze new features like facial expressions and cues.

The second improvement of the data set would be to have complete uninterrupted video of the pitcher on the mound. Video broadcast of games are geared toward the viewing pleasure of fans. This means the broadcast changes cameras to the batter, to the fans in the crowd, or to anything else going on at the stadium. Having an uninterrupted video from the time the pitcher steps on the mound to when he releases the pitch would be very valuable. This would allow us to analyze the entire period the pitcher is in the set position, and to analyze the time before the pitcher comes set.

7.1.2 Expanding Method to All Pitchers

During the 2020 MLB season, 806 pitchers pitched in a game [29]. We only analyzed three pitchers. It would be really interesting to see the rest of the pitchers get analyzed to see if they possibly tip their pitches. This would be a large undertaking, and there

would need to be automated methods to download and crop the video effectively. Though it would take time, the results would be interesting to look at.

7.1.3 Creating an Almost Real-Time Defensive Tool

As we covered previously, using technology to steal signs in real-time is illegal and heavily frowned upon. However, using technology on the defensive side seems like a viable tool. Since a pitching team's only defense to tipping pitches is to recognize the tip themselves, it would be beneficial to have a model analyze every pitch in near real-time to see if the pitcher is tipping their pitches. If the batting team seems to know what is coming, they can use the model to check if the pitcher is tipping pitches. Alternatively, for pitchers that have tipped pitches before or have known tendencies, a model can look for these tendencies and alert the team when it is happening again. If the pitcher is doing well, there is no need to alert the pitcher, but if the batting team seems to have picked up on it, the team can alert the pitcher.

7.1.4 Aid to Film Room Analysis

The last future work would be to create a tool for batters to analyze pitchers in the film room. Using this would not be allowed during the game, however, using it while studying in the film room after or before games is allowed. Dedicated players and staff watch film daily to find a competitive edge. Creating tool to help them analyze film more efficiently and effectively would help them find advantages faster. For example, the tool could alert the batters that the next pitcher they are facing has a tip with something to do with their glove. The batters can look at film and try to see if they can pick it up in real-time. If they feel it is successful, they can take it to their at bat

during a game. Though the tool can help them find a tip, the batter still has to put in the time to be able to recognize it in the film room and while they are batting.

BIBLIOGRAPHY

[1] Cal Poly Github. http://www.github.com/CalPoly.

[2] Darvish stats. https://baseballsavant.mlb.com/savant-player/yu-darvish-506433?stats=statcast-r-pitching-mlb.

[3] Glasnow stats. https://baseballsavant.mlb.com/savant-player/tyler-glasnow-607192?stats=statcast-r-pitching-mlb.

[4] Strasburg stats. https://baseballsavant.mlb.com/savant-player/stephen-strasburg-544931?stats=statcast-r-pitching-mlb.

[5] This tweet shows how rays' tyler glasnow was tipping pitches vs. astros. https://nesn.com/2019/10/this-tweet-shows-how-rays-tyler-glasnow-was-tipping-pitches-vs-astros/.

[6] tyler glasnow 2019 game by game pitching logs. https://www.baseball-almanac.com/players/pitchinglogs.php?p=glasnty01&y=2019.

[7] Moneyball, 2011.

[8] Best way to binge baseball? mlb film room. https://www.mlb.com/news/mlb-film-room-launch, 2020.

[9] Mlb film room. https://www.mlb.com/video/search, 2020. Web site.

[10] D. Bernstein. Astros cheating scandal timeline, from the first sign-stealing allegations to a controversial punishment. https://www.sportingnews.com/us/mlb/news/astros-scandal-timeline-sign-stealing-punishment/zbe6j4yoi1g21ia0say31zv3n, 2020.

[11] J. Brownlee. Difference between classification and regression in machine learning. https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/, 2019. Accessed: [May 2021].

[12] J. Brownlee. A gentle introduction to object recognition with deep learning. https://machinelearningmastery.com/object-recognition-with-deep-learning/, 2021. Accessed: [May 2021].

[13] J. Brownlee. How to develop a random forest ensemble in python. https://machinelearningmastery.com/random-forest-ensemble-in-python/, 2021. Accessed: [May 2021].

[14] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019.

[15] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[16] R. Chen, D. Siegler, M. Fasko, S. Yang, X. Luo, and W. Zhao. Baseball pitch type recognition based on broadcast videos. In H. Ning, editor, *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health*, pages 328–344, Singapore, 2019. Springer Singapore.

[17] O. P. R. Committee. Official baseball rules. https://content.mlb.com/documents/2/2/4/305750224/2019_Official_Baseball_Rules_FINAL_.pdf, 2019.

[18] D. Cournapeau. Scikit-learn. https://scikit-learn.org/stable/.

[19] R. Girshick. Fast r-cnn, 2015.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[21] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[22] K. Hickey, L. Zhou, and J. Tao. Dissecting Moneyball: Improving Classification Model Interpretability in Baseball Pitch Prediction. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[23] P. Hoang, M. Hamilton, H. Tran, J. Murray, C. Stafford, L. Layne, and D. Padget. Applying machine learning techniques to baseball pitch prediction. 01 2014.

[24] F. Massa and R. Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: [April 2021].

[25] W. McKinney. pandas. https://pandas.pydata.org/.

[26] MLB. Statcast. https://www.mlb.com/glossary/statcast, 2020.

[27] A. Piergiovanni and M. S. Ryoo. Fine-grained activity recognition in baseball videos, 2018.

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016.

[29] B. Reference. 2020 mlb standard pitching. https://www.baseball-reference.com/leagues/MLB/2020-standard-pitching.shtml, 2020.

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[31] ryan Plunkett. Pitch type prediction in major league baseball. 2019.

[32] G. D. Sidle. *Using Multi-Class Machine Learning Methods to Predict Major League Baseball Pitches*. PhD thesis, 2017. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-05-18.

[33] L. Sommer. How can anyone hit a 90 mph fastball? science explains! https://github.com/facebookresearch/maskrcnn-benchmark, 2013. Accessed: [May 28, 2021].

[34] D. Soni. Supervised vs. unsupervised. https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, 2018. Accessed: [May 2021].

[35] B. Svrluga. Banning the shift might not fix baseball's problems, but it's worth an experiment. https://www.washingtonpost.com/sports/2021/03/18/baseball-shift-rule-changes/, 2021.

[36] N. Team. About. https://numpy.org/about/, 2021.

[37] O. Team. About. https://opencv.org/about/, 2021.

[38] P. Thompson. Yu darvish now wonders if he was tipping pitches in 2017 world series, or were astros cheating? https://madison.com/wsj/sports/baseball/professional/yu-darvish-now-wonders-if-he-was-tipping-pitches-in-2017-world-series-or-were/article_c10b8dab-583d-5743-abfd-af555ed35817.html#:~:text=Yu%20Darvish%20throws%20during%20the,the%20game%20in%20Los%20Angeles.&text=Yu%20Darvish%20had%20a%20nightmare%

20of%20a%20World%20Series%20in%202017.&text=For%20two%20years%2C%20the%20story,and%20the%20Astros%20took%20advantage, 2020.

[39] J. Turbow. The complete guide to tipping pitches in baseball. https://thebaseballcodes.com/2021/01/27/the-complete-guide-to-tipping-pitches-in-baseball/, 2021.

[40] T. Verducci. How the nationals fixed stephen strasburg and saved their season. https://www.si.com/mlb/2019/10/30/stephen-strasburg-nationals-saved-season-world-series-game-6, 2019.

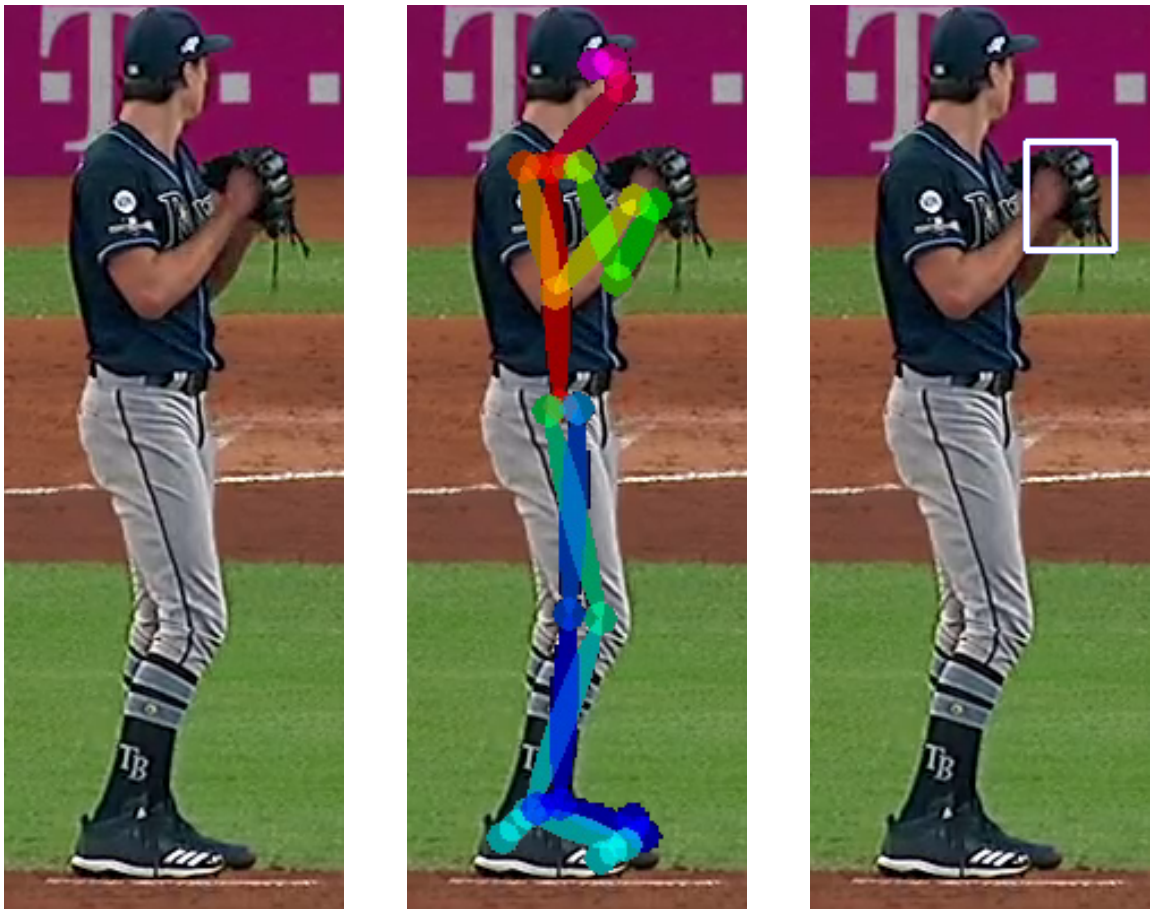
[41] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[42] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2: A pytorch-based modular object detection library. https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/, 2019.

APPENDICES

Appendix A

FEATURE VISUALIZATION

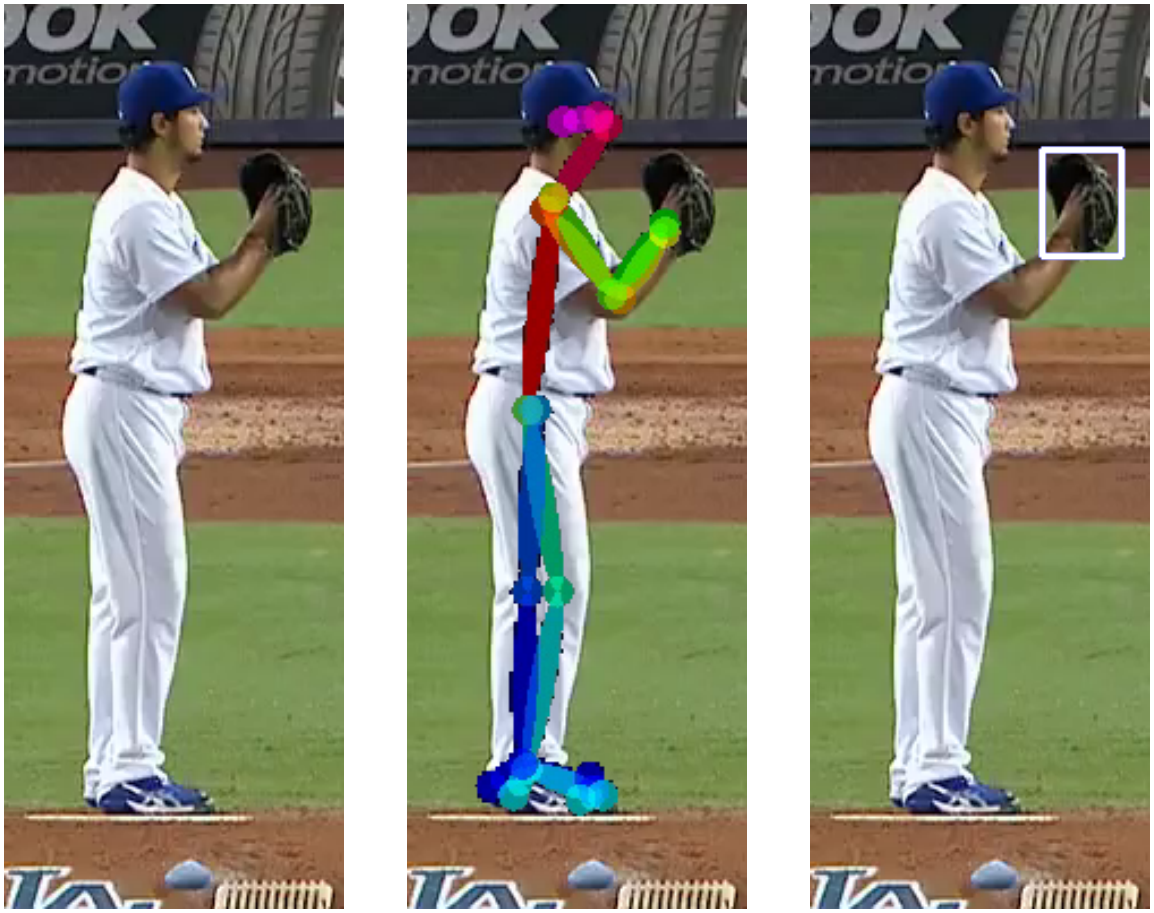


(a) (b) (c)

Figure A.1: Glasnow in the set position (a), with pose estimation overlay (b), with glove bounding box (c)

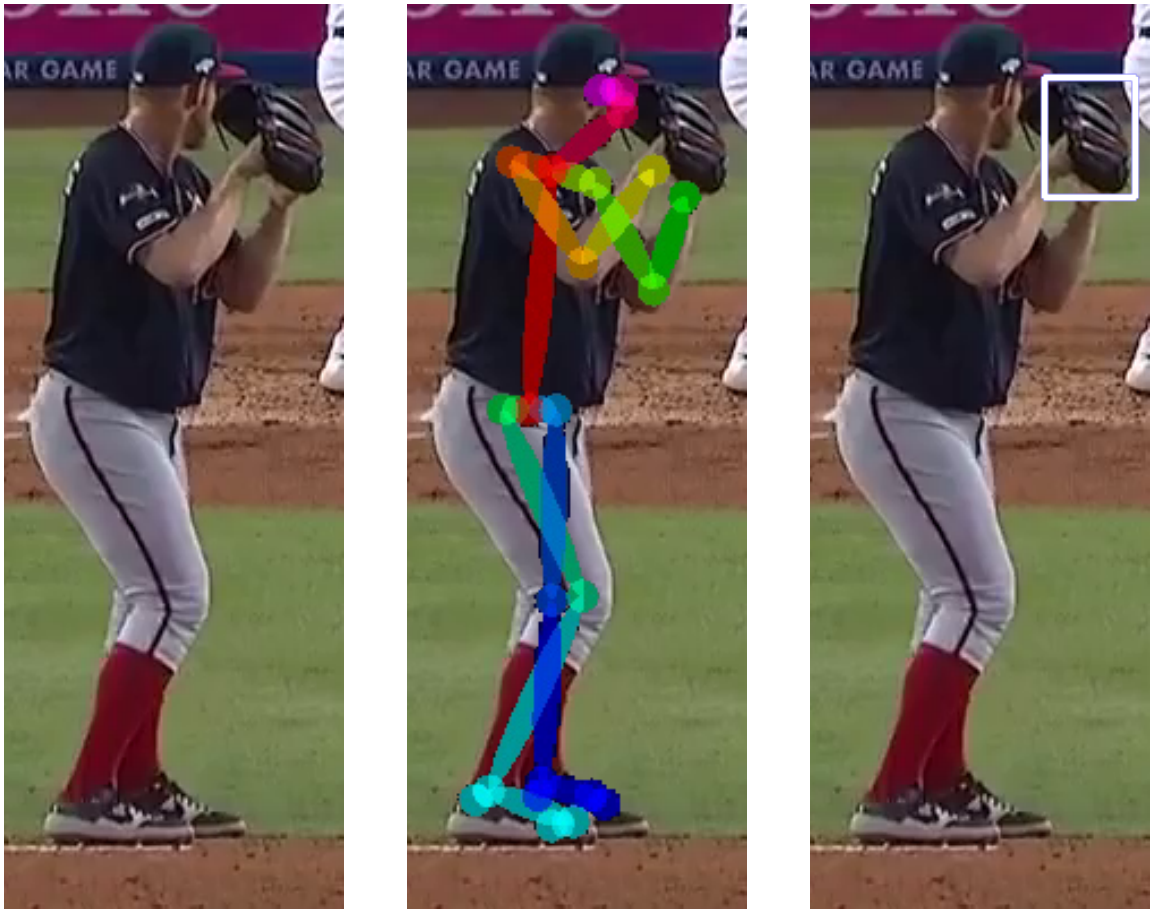
(a) (b) (c)

Figure A.2: Glasnow during his windup (a), with pose estimation overlay (b), with glove bounding box (c)



(a) (b) (c)

Figure A.3: Darvish in the set position (a), with pose estimation overlay (b), with glove bounding box (c)



(a) (b) (c)

Figure A.4: Strasburg in the set position (a), with pose estimation overlay (b), with glove bounding box (c)