Volleyball Trajectory Prediction Using Real-Time Object Detection

Carol Song
The Harker School
San Jose, CA
caroltianyao@gmail.com

ABSTRACT

The goal of this project is to determine where a volleyball will land relative to the player who sent it out. We use the object-detection system of YOLO, a subset of darknet, to detect both the server and the ball in action shots of volleyball. For this, we train YOLO on a custom dataset by having the system learn how to detect a volleyball and a server from images that were pulled off the internet and labeled. Using our own model, we then input our own images of players striking volleyballs where the distance the ball will travel is known. From the data of the bounding boxes produced by YOLO, we are able to use the necessary variables to calculate where the ball will land. The experiment yields successful results as the average percent error for the trials is low. Future projects can utilize these results to build more complex ways to play volleyball with AI or detect objects in the context of sports.

1 BACKGROUND

YOLO (You Only Look Once) [1, 2] is an open-source object-detection system based on darknet that can pick out simple, everyday objects in pictures, videos, and real-time webcams by creating a bounding box around each detected object. To achieve this, the system utilizes Convolutional Neural Networks (CNN) [4] to process the input data. CNN is the name for a type of artificial intelligence that is used to perform a series of operations on the pixels in visual images. These operations are done through many layers of artificial neurons that process the image by extracting more and more complex features of the image, starting with the edges of the photo until it can identify the specific objects in it. YOLO's system function can be broken down into three procedures: residual blocks, bounding block regression, and intersection over union (IOU). Each image or video frame is split into several grids so that each cell, or residual block, has its own inner object to detect. If an object is recognized, a bounding box is drawn around the entire perimeter of it and the object name, percent detection accuracy, and coordinates of the box are outputted. IOU is a value that compares the real box of the object to the bounding box, and it is employed to eliminate the bounding boxes that do not match the real box so the output can be as accurate as possible.

The standard, pre-trained model of YOLO can detect 80 common objects, such as dogs and bicycles. However, users can also train the system to detect custom objects of their own choice. YOLO has been used to accomplish a multitude of projects before. For example, the object detection system has been used to detect roadside traffic signals such as stop signs and yield signs and alert the driver of them. It has also been used to detect people in a building so that the total number can be counted. For this project, we only need to train the model to detect 2 objects, balls and servers, which we will explain the reason for in the next sections.

2 INTRODUCTION

The purpose of this project is to use the YOLO system to predict where a volleyball in projectile motion will land relative to the initial location of the ball. To do so, example images of players hitting balls must be used and from them, 3 data points must be extracted: the height, the angle, and the speed at which the ball was hit. Inserting these values into the equation for trajectory motion will yield the distance the ball has traveled from its initial position, or in other words, its displacement. The results of this experiment can benefit future projects involving artificial intelligence and volleyball. For instance, further developing this model can allow robots to do advanced things like play volleyball.

In Section 3, we briefly introduce the variables necessary to conduct the experiment. In Section 4, we dive into exactly how the custom YOLO model was trained to detect the ball and the server in each image. In Section 5, we outline the methods and algorithms used to calculate the final value. In Section 6, we show a couple of examples of trial runs, as well as both the data and results from them. In Section 7, we analyze our results and discuss what possible experimental errors that may have occurred.

3 METHOD PREFACE

To accomplish the goal, we take videos of a player striking or passing a volleyball from the side and record the exact distance between the landing spot and myself, in meters. This value is the theoretical, or expected value, of each trial. Then, from each video, we hand pick two frames. The first

frame is the moment closest to after the ball is contacted, and the second one is the frame right after. In order to derive the distance in which the ball will travel, we have to first obtain three variables: the height in which the ball was hit, the angle in which the ball was hit, and the velocity at which the ball was hit. Thus, to figure out these numbers, we have to train YOLO with a custom dataset to detect both volleyballs and the person sending out the volleyball.

4 MODEL TRAINING

We download around 50 images containing the volley ball and servers from the Internet. In order to help the system understand what exactly it should be detecting, we use OpenLabeling [3], an open-source image and video labeling tool. To label every picture, we draw boxes around each volleyball under the class name, "ball", and boxes around each server under the class name, "server". Next, we set up custom configuration files in a “cfg" folder so that the variables could match up to our custom dataset. In the .cfg file, we have modified variables including the line batches and subdivisions. We download the custom weights file to put in the custom cfg folder. To run darknet, we open a new terminal and inputted the command:

```
$ ./darknet detector train custom_cfg/
custom.data custom_cfg/yolov4-custom.cfg
custom_cfg/yolov4.conv.137 -map.
```

The training process ran for 40+ hours, in which an mAP is produced from the results. The mAP, short for mean average precision, is a chart that outlines the training progress by computing the percent accuracy for each iteration of training. In the mAP produced for our training, the model hit 100% accuracy in just the 1,400 iterations. The percentage continued to fluctuate as a result of possible over-training but we allow to model to run until all the iterations were complete.

After the system is successfully trained, we run our own picture frames into the trained model. Instead of calling the standard configuration and weights files that YOLO provides, we call the custom ones instead and received the bounding boxes for each volleyball and server. YOLO automatically outputs data along with the box, and the applicable ones are the height of box around the ball, the height of the box around server, the distance from the top of each box to the top of the frame (y-coord), and the distance from the left side of each box to the left side of the frame (x-coord), each of these given in the unit of pixels.

The computer we use for training has an Intel i7 CPU, 16GB RAM, and an NVIDIA GeForce GTX 1660 Ti GPU. The operating system is Windows 10.

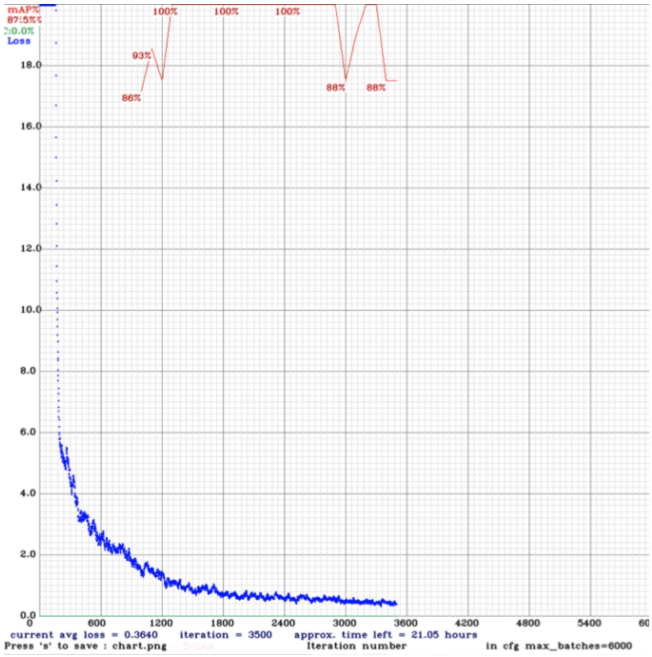

Figure 1: mAP for the custom model training

5 PREDICTION METHOD AND ALGORITHM

The first relationship established in the code was the ratio from pixels to meters, which is obtained by measuring the diameter of the ball used (around 0.2 meters) and the height of the bounding box, which is the diameter of the ball in the picture in pixels. We will be referring to the bounding box around the ball as the ball-box and the bounding box around the player as the player-box.

To determine the height in which the ball was sent out, we add the y-coord and height of the player-box from the first frame, which gives the distance between the ground in the image to the top. Then from this value, we subtract the y-coord of the ball, which generates the distance between the ground to the top of the ball-box. The last step is to subtract half of the height of the ball-box, which gives us the distance from the ground to the center of the volleyball, which is the height that we are looking for.

Next, we have to figure out the angle in which the ball was sent. We merge each set of frames into a single picture in which we ran through YOLO, so that the system detects both the balls as well as the player. We draw a right triangle on the picture, with the centers of each ball the vertices for the hypotenuse. For each ball-box, we solve for the center by adding half of the diameter of the ball (in pixels) to the x-coord and to the y-coord. Then, we determine the length of the legs of the right triangle by subtracting the values of

each pair of center coordinates. The angle in which the ball is stricken will be the arc-tangent of the vertical leg over the horizontal leg.

Lastly, we determine the velocity at which the ball traveled. Velocity is determined by the distance the ball traveled by the time it took to travel the distance. Obtaining the time was simple: it is the time (in milliseconds) between the two frames. To figure out the distance, we use the Pythagorean theorem on the length of the legs of the right triangles that was determined from the previous step. This gives the length of the hypotenuse in pixels, which we convert to meters using the ratio from the diameter of the ball. Thus, we are able to get the velocity in meters per millisecond.

To determine the distance in which the ball traveled, we used the formula:

$$d = v \cdot \cos(a) \cdot (v \cdot \sin(a) + \sqrt{(v \cdot \sin(a))^2 + 2 \cdot g \cdot h})/g$$

where d is the distance the ball travels or its displacement, v is the velocity of the ball, g is the gravitational constant, which is equal to around $9.8 m/s^2$.

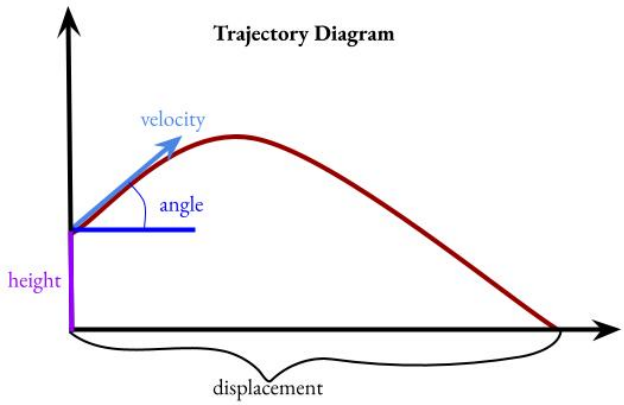

Figure 2: Diagram representing the trajectory path of an object

We wrote the equations into the Python editor with methods for each of the three variables and a dictionary for the data from the bounding boxes which is used in every method. This code was used for every trial that was conducted.

6 EXAMPLES

Figure 3 is an example of a trial run using our custom trained YOLO model. The position of the ball right after contact and the next frame are clearly incorporated into the picture so that YOLO can detect them with ease. In this figure, both the balls and the person are detected by the system, with a bounding box drawn around all of these objects, although not perfect.

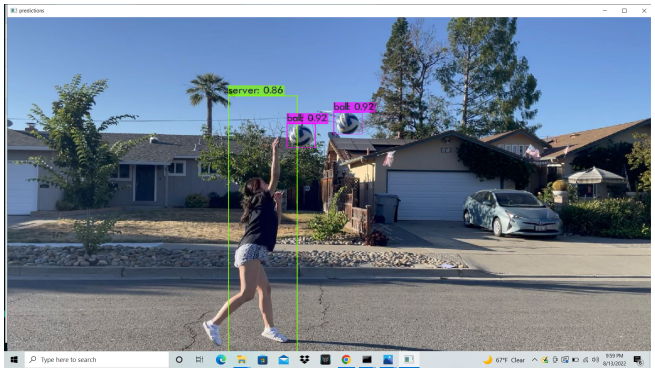

Figure 3: Custom Trained Example 1

Figure 4 is another example of a trial run using our custom trained model. The position of the ball right after contact and the next frame are clearly incorporated into the picture so that YOLO can detect them with ease. However, in this figure, system has detected only one ball. The person is also detected, but there are leaves at the top of the frame that are also detected.

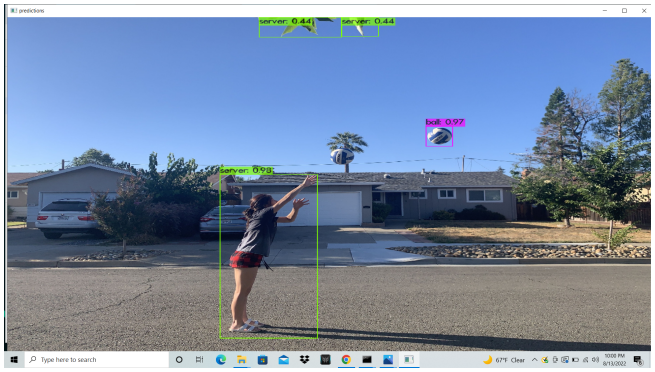

Figure 4: Custom Trained Example 2

As shown from the examples of images that were run through our trained model, the bounding boxes around each detected object are still far from precise and not all of the necessary objects were detected. In order to get maximum accuracy, we ultimately use the pre-trained YOLO model to conduct 10 trials as it is far more precise. Figure 5 shows the bounding boxes that are drawn for the image of Figure 3: Custom Trained Example 1. Note how the edges of the bounding boxes are almost perfectly wrapped around the object.

Table 1 depicts both the experimental and theoretical values of the ten trials that were conducted, as well as the percent error for each. Out of the ten, seven trials yielded values smaller than the expected value. Only trials 3, 8, and 10 resulted in values greater than the actual value. The trial

Figure 5: Pre-Trained Example 1

that yielded the best result is trial number 8, with a percent error of just 0.80%.

Table 1: Results of 10 Trials

Trial	Predicted Value (meters)	Actual Value (meters)	Percent Error
1	10.64	10.92	2.56%
2	7.67	8.74	12.21%
3	9.14	9.02	1.41%
4	9.96	10.16	2.00%
5	14.86	15.16	2.01%
6	9.14	10.77	15.09%
7	11.48	11.91	3.62%
8	12.72	12.62	0.80%
9	12.11	12.98	6.65%
10	12.09	10.92	10.70%

7 ANALYSIS

Noticeably, the bounding boxes drawn around both the balls and the player are not perfectly precise in both the examples. In example 1, the percent error is 2.56% and in experiment 2, the percent error is 12.21%. Both bounding boxes for the server does not exactly line up with the feet, which causes the initial height of the ball to be derived as greater than it should be. This is the reason why we ultimately used the pre-trained model of YOLO to test our algorithm. Still, the predicted values were not completely precise, most being smaller than the actual value. Experimental errors that may have occurred include the wind and the camera angle. The wind, depending on its direction, could have pushed the ball away from its initial position, causing the value to be greater than it should be, or towards, which decreases its value. If the ball was not sent at an angle perpendicular to the camera, the results will vary from the theoretical value as well. If the camera was slightly to towards the back, then the angle would have been measured as greater than it should be. Thus, since the angle is proportional to the distance in the equation, the experimental value would have been calculated to be greater than it should be as well. Vise versa, if the camera was held slightly more to the forward of the server, than the calculated distance would have come out less than it should be, which is likely something that had happened in this experiment. Still, the experiment is quite accurate; the average percent error of all ten trials conducted is around 5.71%.

8 CONCLUSION

The purpose of this experiment was to determine where a volleyball in projectile motion will land relative to its initial position. We utilized= the YOLO system to detect the necessary objects in video frames to provide the data needed to conduct the experiment. To improve the results of this experiment, we can train YOLO on a custom dataset with more images. The process will take much longer but yield more accurate detections. Thus, we can complete the project using our own model instead of the pre-trained one. In the future, we can expand on this project by using our model to detect and predict in real-time. This information could be used to teach robots how to play volleyball or create volleyball training machines that are more efficient. The code used for this project is available on Github at “https://github.com/carolsonggg/YOLO-volleyball".

9 ACKNOWLEDGEMENTS

This project was completed during a summer internship under Professor Zhu Han of Electrical Engineering at the University of Houston. A special thank goes to my parents who helped me take the videos and measurements needed for the trials and supported me throughout the process.

REFERENCES

[1] [n.d.]. Darknet. https://github.com/pjreddie/darknet.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *CoRR* abs/2004.10934 (2020). arXiv:2004.10934 https://arxiv.org/abs/2004.10934

[3] J. Cartucho, R. Ventura, and M. Veloso. 2018. Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2336–2341.

[4] Rahul Chauhan, Kamal Kumar Ghanshala, and R.C Joshi. 2018. Convolutional Neural Network (CNN) for Image Detection and Recognition. In *First International Conference on Secure Cyber Computing and Communication (ICSCCC)*.