Convolutional Neural Networks for Ball Tracking and Sports Analytics

Claire Lorenzo
clorenzo@mit.edu

Fabián Yáñez-Laguna
fyanez@mit.edu

Maxwell Zetina-Jimenez
zetina@mit.edu

Abstract

Ball tracking has been successfully implemented in the past, as it can be used to project ball trajectory and yield useful statistics such as velocity. To enhance the extraction of sports analytics from video inputs, we provide both velocity and acceleration graphs of the ball's motion without the use of multiple cameras or prior knowledge of environment dimensions. To achieve this, we use an object detection model, a depth-estimation model, and projective geometry for distance measurement. We apply this end-to-end pipeline to a wide range of input videos, including personal training videos and video footage of publicly available sports games, primarily serving day-to-day users who want to analyze their sports performance without the need for private, expensive complex systems. We show how existing object detection models can be paired with real distance estimation models to accurately produce useful analytical insights such as ball velocity and acceleration without requiring more than one camera, as in the case with most ball-tracking computer vision systems.

1. Introduction

In the field of sports analytics, real-time ball tracking is a crucial feature of video analysis for coaches, players, and analysts. In recent years, the use of technology has enhanced sports statistics to track data and evaluate players. Our project aims to maximize the effectiveness of ball-tracking models in casual sports analysis, and we focus on accurately tracking the ball's trajectory to calculate ball velocity and acceleration. Specifically, we do this with a single camera and can apply it to arbitrary input videos, as opposed to the majority of ball-tracking systems that require multiple fixed cameras and constraints on the types of videos it can analyze. This choice is motivated by the current robustness of existing deep-learning models in detecting objects with the use of convolutional neural networks, as well as the fact that most amateur athletes and regular users cannot afford very complicated sports analytics systems. Our project hopes to build on this and use the detection data to create statistics, since most of the current algorithms are focused on mainstream sports industry videos. The metrics our model outputs provide valuable insights into understanding game dynamics, player performance, and skill, thus contributing to improving sports analytics and the improvement of athletes' capabilities.

There exist many models that focus on object tracking, velocity calculations, and other metric estimations. However, we aim to create a generalized end-to-end pipeline that provides multiple analytical insights from arbitrary, more casual input videos. Pairing YOLOv5 — a state-of-the-art object detection model — with a depth-predicting model enables us to generate results on velocity and acceleration of balls. These models are independent of each other, allowing us to employ them sequentially in our pipeline. YOLOv5 already contains a velocity prediction mechanism. However, these predictions are often inaccurate, and our system is more robust due to its ability to provide both velocity and acceleration while also taking depth into account with a model specialized for depth prediction.

Since the two models we employ have data dependencies with each other, they must be run sequentially. To improve performance through parallel computing, we would have to train a model from scratch and produce analytical insights from a single run through the video, as opposed to reading the video input three separate times for each module of the pipeline. However, we prioritize accuracy over performance, thus justifying the use of a sequential pipeline.

2. Literature Review

2.1. YOLO

Several ball detection algorithms already exist. In particular, we base our work on one specific object detection algorithm, YOLO, which can be used for object detection tasks in general. For our purpose, we change the model parameters to detect balls in particular. Compared to other real-time object detection systems, YOLO has one of the highest Mean Average Precision while also being very computationally efficient, which justifies our decision to use this model. The model's architecture consists of 24 convolutional layers, 4 max-pooling layers, and 2 fully connected layers. YOLO has several versions, but we chose to experiment with two: YOLOv5 and YOLOv8, the latter of which

has a slight edge in terms of precision and recall; however, it sometimes detects non-existing objects, which does not occur as often with YOLOv8. For the first phase of the pipeline, which involves object detection, we determined YOLOv5 is sufficiently accurate based on its performance on a batch of test images we chose. Therefore, we chose to use YOLOv5 due to its slight edge in performance, which is critical when dealing with videos consisting of dozens or even hundreds of frames.

2.2. Hawk-Eye

While we focus on applying the model to pre-recorded videos, the use of real-time ball tracking has been successfully implemented, such as in the case of Hawk-Eye [8], a state-of-the-art computer vision system used in many different sports to precisely track the trajectory of a ball and provide statistics on it. However, this optical tracking technology relies on the use of 6 high-performance cameras, allowing it to utilize triangulation to project the path within 4mm of the true path. Additionally, this system can use knowledge of court or field dimensions of the field of vision to effectively predict distances traveled by the ball. We aim to provide a ball-tracking model with the use of a single camera and no prior information on environment dimensions, allowing users to input a pre-recorded video and get statistics on the ball trajectory. While there there are models that perform ball-tracking on input videos, they do not provide statistics on the trajectory of the ball. We address the gap of the use of an analytical ball-trajectory model for videos that are pre-recorded and without the use of multiple cameras filming from various angles. While we do not expect the accuracy of the model to be as accurate as computer vision systems such as Hawk-Eye due to lack of cameras and recording quality, we address the need for a ball-trajectory model that estimates statistics on videos that are pre-recorded without any prior knowledge on the dimensions of the environment being recorded.

2.3. Related Works

Other related works include soccer ball tracking algorithms, trajectory prediction algorithms, and color-based segmentation and feature detection for applications in cricket and football. Other models are based on different convolutional neural networks, such as Zhang's *Efficient Golf Ball Detection and Tracking Based on Convolutional Neural Networks and Kalman Filter* [3]. We aim to use existing networks and algorithms to perfect the accuracy of velocity predictions based on existing object detection techniques. To accurately track distance traveled, we also require the use of a monocular depth estimation model, which we explain more in detail in the Approach section.

3. Approach

3.1. End-to-End Pipeline

To generate velocity and acceleration plots of the ball, we require information on the change in position of the ball over time. To calculate the change in position, we use the object-detection model, YOLO, to create a bounding box around the ball, which is used to find the x and y pixel coordinates. The other model we use, pytorch-mono-depth [10], estimates the depth of the ball. With this information, the 3-dimensional world coordinates of the ball can be predicted for each frame. Finally, the change in coordinates, paired with the known frame-per-second rate of the camera, are used to calculate velocity and acceleration over time. An overview of this process is shown in Fig. 1.

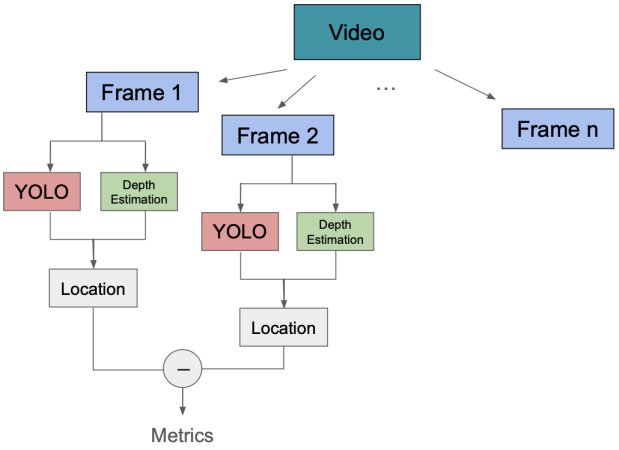

Figure 1. **System Overview**

3.2. Methodology

To track the change in position of the ball over successive frames, we first use YOLOv5 to detect the ball as in Fig. 2. We identify the center of the bounding box of the ball in each frame, as this corresponds to the center of gravity of the ball. The x and y camera coordinates of the ball are not sufficient to accurately predict distance traveled. To precisely track the distance traveled, we must consider 3-dimensional space. To do so, we use projective geometry to translate the 2D camera coordinates to the 3-dimensional world coordinates of the ball, which can then be used to calculate 3-dimensional displacement. However, we need information on the depth of the ball to do this. We utilize a model named pytorch-mono-depth, an implementation of a monocular depth prediction model, to estimate the depth of the ball. Thus, the world coordinates of the ball in each frame are:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = Z \cdot \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} \quad (1)$$

where Z is the depth, x_{pix} and y_{pix} are the x and y pixel coordinates of the center of the ball, p_x and p_y are the pixel coordinates of the center pixel, and f_x and f_y are the camera's focal length dimensions.

Given the change in 3D world coordinates, we calculate the Euclidean distance of the ball's location between frames, which can then be used to calculate velocity as follows:

$$v = \frac{d \cdot fps}{2} \quad (2)$$

where d is the predicted distance traveled by the ball in meters between the two frames and fps is the frame rate of the camera. The calculation for acceleration follows trivially given the velocity between frames. Finally, our system provides the average velocity and acceleration and generates plots of the velocity and acceleration over time.

Figure 2. **Ball Detection in an Image** YOLOv5 attempts to detect the ball and forms a bounding box around its location.

Note that at high speeds, the ball's bounding box can be distorted due to the ball appearing stretched out in the direction of movement in every frame, as demonstrated in Fig. 3. This does not affect the accuracy of the ball detection since the center of gravity always remains at the center of the bounding box. Additionally, since the quality of the video can cause the model to mislabel an object, we account for frames in which the model's bounding box is inaccurate by filtering out the outliers and continuing to keep track of frames with a valid ball detection.

3.3. Experiments

To test, we utilized a combination of personal videos we filmed and publicly available videos from datasets such as SoccerNet, NBA replays, and tennis footage. These datasets allow us to compare the difference between the model's predicted ball trajectory statistics with ground truth statistics. While there is not a shortage of data, these datasets have several limitations. Namely, they lack a variety in camera angles and contain partial occlusions that can alter calculations such as position changes when performing velocity calculations. Thus, to augment these datasets, we filmed carefully planned videos of our own. To test the robustness of our pipeline, we filmed many different types of balls varying in size and color, including a baseball, soccer ball, and ping pong ball. In particular, this allowed us to test how well the object detection model can detect small balls from a distance and the cases in which the balls blend in with the background.

3.3.1 Experiment 1

First, we tested on one of our personal videos that we took indoors with no other objects in the frame. This served to test the basic functionality of our model: the ability to accurately identify the ball in each frame and calculate velocity and accleration.

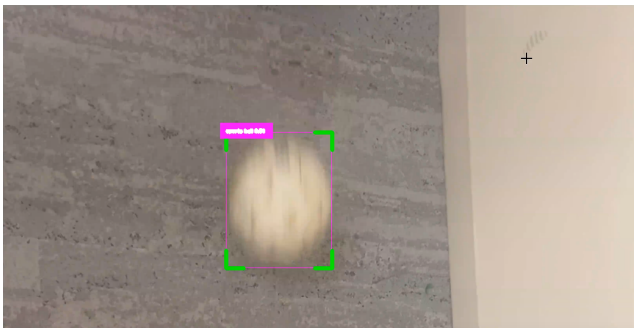

Figure 3. **Ball Tracking in a Video** YOLOv8 can track a moving ball in individual video frames. This can be extended to analyze the change in the ball's location, even with apparent ball distortions due to the captured movement in individual frames.

3.3.2 Experiment 2

Next, we tested our model on a set of videos filmed outside with a fixed camera angle. Each video had a different type of ball, varying in size and color. This experiment aimed to test our system's ability to calculate velocity and acceleration in videos with a lot of noise, such as other objects present and non-uniform backgrounds that blend in with the ball.

3.3.3 Experiment 3

Our final experiment used video footage from publicly available sports footage from professional games or exhi-

bitions. This experiment served the purpose of determining how our system generalizes to more professional settings with less constraints on the filmed environment and camera angles.

4. Results & Future Extensions

4.1. Accuracy

To quantify this performance of our system, we chose to use mean squared error, summing over the differences of all test videos values obtained running through the pipeline and the "actual" values which we calculated thanks to our physics trajectory, which we know is very close to accurate:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3}$$

where Y_i and $\hat{Y}_i$ are the true metric and the predicted metric, respectively. Addtionally, we individually visually analyzed the accuracy of each video to assess the shortcomings and strengths of our computer vision system.

4.1.1 Experiment 1

For our first experiment, our model performed extremely well. As described, we used personal videos filmed indoors with no other objects or noise in the background to evaluate the basic functionality of our model. These videos include different camera angles and ball launch angles. To test the accuracy, we used calculated the true velocity using basic projectile motion physics principles, comparing these results to the predicted velocity and acceleration values.
For instance, the tracking of a medium-size ball thrown vertically in the upwards direction are shown in Fig. 4.

As seen in Fig. 4, the velocity drops to 0 after 5 frames, which is when the ball reaches the top of its trajectory. The ball then accelerates until it abruptly becomes 0 when it lands.

We generated a frame-by-frame velocity calculation based on a ball being thrown upwards at a vertical velocity of 5 m/s, and we got a MSE of 0.195. We got similar mean squared errors for the diagonal and horizontal throws and kicks, which we deem to be satisfactory results given our resources.

4.1.2 Experiment 2

For our second experiment, we test our model on a set of personal videos that we filmed outdoors, with a more noisy background and a variety of other objects in the frame. For instance, we filmed a baseball being thrown, two people passing a soccer ball to each other over a large distance, and two people playing high-speed ping pong.

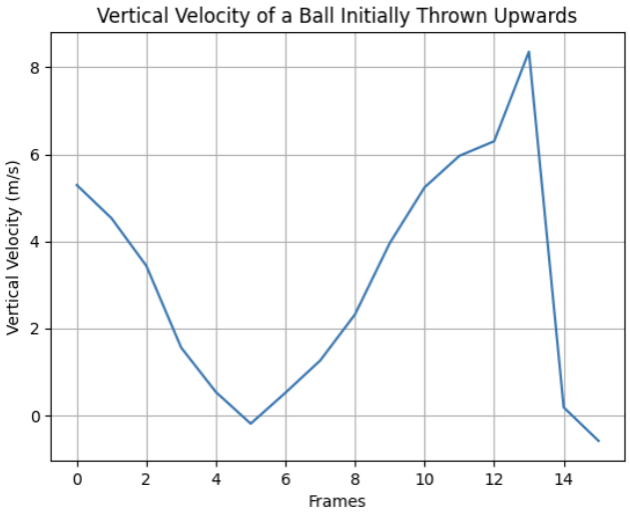

Figure 4. **Velocity Calculations of Our Model for an Upwards Ball Throw** Curve shows decrease of velocity until 0 as the ball goes up and then increase as the ball falls back down.

For the video in which two people throw a baseball, we got the following results shown in Fig. 5.
For this second experiment, we estimate the throw velocity to be about 25m/s, yielding a squared error of approximately 0.2. When comparing this to YOLOv8's generated speed value, which does not give frame by frame value, we find that our results give a higher precision and more detailed view of how the velocity trends over the frames, and we also provide acceleration.

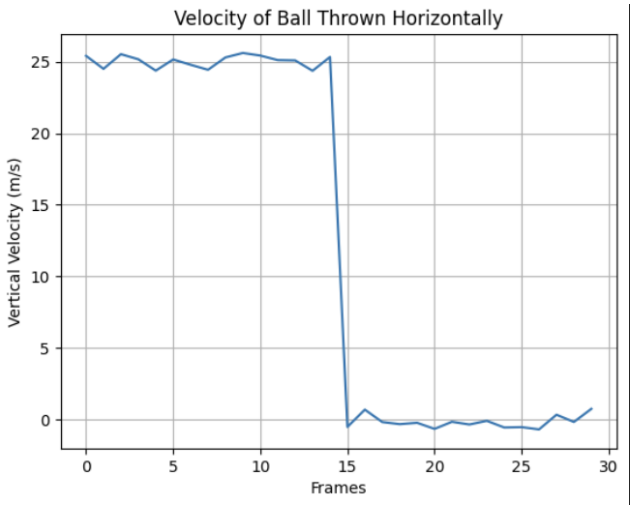

Figure 5. **Velocity Calculations of Our Model for a Baseball Ball Throw** Curve shows constant velocity until ball is caught.

4.1.3 Experiment 3

In this experiment, we applied our pipeline to a variety of nationally televised sports games. As expected, due to the amount of noise and changing camera angles, our system performed very poorly. In many of the frames, YOLO was not able to detect the ball, causing the velocity plots to oscillate rapidly. In the videos in which the model detected the ball in enough frames to calculate velocity and acceleration, the accuracy was very low. These results were expected since our system is not meant to handle changing camera angles and videos in which the ball is obstructed in some frames.

(a) Ball Detection in Real Game Replay

(b) Failed Detection

Figure 6. **Ball Detection in Subsequent Frames** YOLO failed to detect the soccer ball in subsequent frames, leading to faulty velocity calculations between frames.

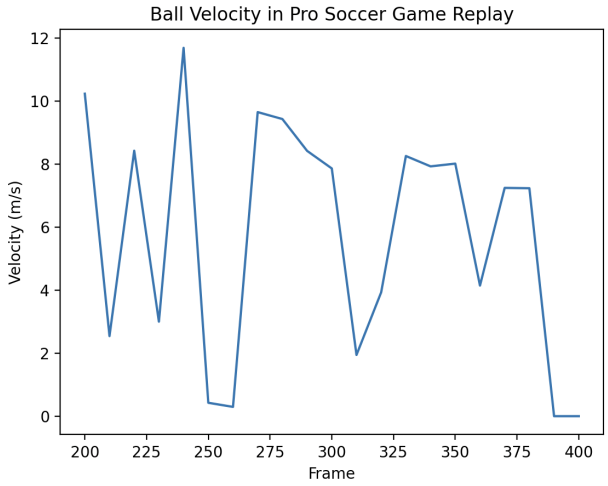

Figure 7. **Velocity of Soccer Ball in a Real Game** YOLO failing to detect the ball in many frames causes a lot of chattering, or oscillations, for the velocity.

4.2. Limitations

While the results were promising for very clear test videos that we took, our system had several key limitations. Firstly, the performance in terms of runtime and complexity of our system was not great compared to other similar models. The main reason for this is because we apply two different convolutional neural networks to each frame. Additionally, due to how the object detection model, YOLO, functions, the performance was also severely affected by the fact that the model detects a variety of objects, but we ignored any object detected that was not a ball. Training a model to solely focus on the task of detecting balls would have improved performance. Despite it being a one-pass system, only reading the input once, the latency is very high. To improve this, we would have to train a neural network from scratch that can detect objects and determine their depth with a single pass while using the small changes in object location to determine the distance traveled. However, given the time limitations of this project, this was not feasible.

Another major shortcoming was the model's inaccuracy when it came to ball movement away from the camera. This was due to the lack of accuracy when determining the depth of the ball, significantly decreasing the accuracy of the velocity and acceleration. Our model was only precise with videos in which the ball only travels horizontally and vertically with respect to the camera, as opposed to moving farther away from the camera.

Our model's inability to detect very small balls was also an issue, as it completely failed to detect the ping pong balls unless they were very close to the camera and in front of a uniform background. More generally, our video struggled to accurately detect the balls in videos with a lot of noise, such as other random objects that the model inconsistently identified as balls in different frames, but this is not as much of an issue for the types of videos we are aiming our project towards.

5. Conclusion

We built this computer vision system with the goal of providing reliable option for a more widespread alternative to complex sports analytics systems that require several cameras. Our model combines existing ball tracking algorithms with depth estimation models to get accurate tracking, velocity and accuracy estimations for daily use from amateur to professional sports players to improve their game. Our model achieves good accuracy, and in general better accuracy than plain tracking systems such as YOLOv5, for most amateur-level indoor and outdoor videos, which is our main purpose, but performs not as well on in-game sports videos due to certain flaws in the object tracking algorithm. For future extensions of our project, we would like to extend our pipeline and methodology to be able to accommodate for more complex videos, smaller objects, and add extra features to our model such as providing additional metrics.

6. Individual Contributions

Each of the three members on our team was responsible for the execution, analysis, and write-up of one of

the experiments. Fabián Yañez-Laguna worked on experiment 1, Claire Lorenzo did experiment 2, and Maxwell Zetina-Jimenez was in charge of experiment 3. The brainstorming and research on related work was mostly done by Claire Lorenzo and Maxwell Zetina-Jimenez, while Fabián Yáñez-Laguna worked on tying together the methodology and math used.

7. Key Resources

Object Detection Model: https://github.com/ultralytics/yolov5

Depth Estimation Model: https://github.com/simonmeister/pytorch-mono-depth

References

[1] Hussain, Shahnawaz. Jadhav, Rohan. Manthalkar, Rutvik. Cricket and Football Detection Using YOLOV5 Algorithm. Research Gate. May 2023

[2] Fitriana, A; Mutijarsa, K.; Adiprawita, W. Color-based segmentation and feature detection for ball and goal post on mobile soccer robot game field. ICITSI. 2016

[3] Zhang, Tianxiao. Efficient Golf Ball Detection and Tracking Based on Convolutional Neural Networks and Kalman Filter. Cornell University. Dec 2020.

[4] An Effective and Fast Soccer Ball Detection and Tracking Method. www.researchgate.net/publication/220931148. Accessed 5 Apr. 2024.

[5] Tohidi, Mohammad. "'Velociball': Calculating Soccer Ball Speed Using Yolo Object Detection." Medium, Medium, 15 Aug. 2023, medium.com/@the.mohammad.tohidi/velociball-calculating-soccer-ball-speed-using-yolo-object-detection-d528c17397ef.

[6] Yang, Yoseph, et al. "Ball Tracking and Trajectory Prediction System for Tennis Robots." OUP Academic, Oxford University Press, 13 June 2023, academic.oup.com/jcde/article/10/3/1176/7197436.

[7] Tan, Ren Jie. "Breaking down Mean Average Precision (MAP)." Medium, Towards Data Science, 2 Mar. 2022, towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

[8] Hawk-Eye Innovations.https://www.hawkeyeinnovations.com

[9] YOLOv5 https://github.com/ultralytics/yolov5

[10] pytorch-mono-depth https://github.com/simonmeister/pytorch-mono-depth