Ball Localization and Tracking in Video

Brett Fazio
University of Central Florida
brett.fazio@hotmail.com

William Chen
University of Central Florida
wchen6255@outlook.com

Abstract

Our project tackles the problems of both localization and tracking in a video, specifically we wish to be able to track a basketball. Our proposed solution leverages the YOLO (You only look once) [3] real-time object detection system in combination with even higher speed object tracking systems to get high performance object object tracking even on a CPU.

1. Introduction

Ball tracking is a common feature in sports entertainment and analytics. Its application in the former field helps viewers engage with the competition and the content, helping them understand the more intricate aspects of the game being played. The latter is a budding industry starting to cater to both professional and amateur sportsmen alike. Never before has one been able to quantify and analyze so much of their skills, such as proper posture and positioning, and their effects on a player's interaction with the ball. Better shooting posture leads to better accuracy, better positioning leads to more opportunities to receive the ball, and so forth. For most sports, the ball remains the focal point for the application of these skills and remains a constant presence while the skills may vary between sports. As such, our study will explore numerous ways to track sports balls throughout single camera videos.

2. Materials and Methods

While each specific implementation may vary, our general approach to ball tracking remained the same for each method we explored. We split ball tracking into two main steps. The first step is to find the ball within the video. The second step is to track it throughout the video.

Every method was implemented in Python 3, which is well supported by a melange of libraries and guides for its applications in computer vision. Deep learning methods were written in the Pytorch framework, while general pipelining was accomplished with OpenCV (along with Numpy and pandas).

While we explored contour-based tracking, we found the results to be poor. Instead, the two main tracking/localization solutions available in our code are YOLO and YOLO in combination an additional tracker. All methods are discussed below.

2.1. Training on Contours

Our initial approach was to detect contours in each frame of the video and feed that data into a neural network for training.[4] The trained network would then be able to track balls throughout a video.

There were however, several issues with this approach that led us to conclude that it was not worth exploring fully. The first of which is that the initial detection was looking for the highest contour "blob" in each frame. While this approach would work well in a sport like volleyball where the ball is consistently in the air, it was unsuitable for sports like basketball and soccer where the ball generally remains below shoulder level. The second issue was that there was no way to programmatically label the output blobs as a ball or not a ball, which therefore led to straining hours of manual labeling for each frame in each video. Finally, the trained network did not perform very well even tested on the video it trained on. While we attempted to alleviate this by having it consider the blob trajectory and path, there was no clear improvement in performance that made the approach worth considering.

2.2. YOLO Detector Overlap Tracking

Before going into our approach, we will begin with a brief overview of YOLO. YOLO works by applying a single neural network to the whole image, then dividing the image into regions, predicting bounding boxes, and then weighting these boxes by probability - this has the advantage of using context within the image (since the whole image is applied) and also only uses a single evaluation making it up to 100x faster than Fast R-CNN. [3]

This tracking method we came up with is relatively simple approach that utilizes overlap. For this, we run YOLO on every frame to localize all objects. We go through them

and find the one of the right category (in this case "Sports Ball") that has the highest overlap with the bounding box from the previous frame. We compute this overlap using IOU (intersection over union).

While the results of this method are discussed in the Results section of this paper, at a high level the method works well if the basketball is clearly identifiable on every frame (no going behind a person's leg or through a net) however it has the side effect of being very slow (it runs at high speed on a Titan X but not on a laptop CPU) compared to the YOLO + Tracking method discussed next.

2.3. YOLO + Tracking Algorithm

We could consider YOLO + Tracking to be our main approach as in our testing it works well and also runs at near real-time speed even on a laptop CPU (once the ball is initially localized).

This approach works similarly to the overlap approach at the beginning, we repeatedly run YOLO on frames until a ball is found - it is possible the ball is covered or is too blurry the first few frames so running until one is found is necessary. Once the ball is localized, the two methods diverge.

Once the ball is localized we begin the "tracking" part of this approach. This entails optimization of the bounding box size (discussed in Hypter-parameter Tuning) and selection of a tracking algorithm. We get these tracking algorithms using OpenCV, specifically the Tracker class which has child classes for all sorts of tracking algorithms like CSRT, GOTURN, MOSSE, and others. Our investigation mostly focused on GOTURN and CSRT - with CSRT performing the best - but that is discussed in their respective sections below.

After the ball is localized and the tracking algorithm is selected, we continue reading in the frames of the video. On each frame we pass the previous bounding box of the ball to the tracking algorithm which very quickly provides a new estimated bounding box. We do this until the box is lost (goes off screen, goes behind a person) or the video ends.

2.3.1 Backwards Tracking

Often times YOLO is unable to detect the ball in the first few initial frames because it may be attached to something (such as a hand) or partially hidden/out of view. To combat this issue, we also run the tracking algorithm on the video backwards, from the frame where the initial ball detection is to the first frame of the video. This will slightly increase the runtime of the program while preventing live tracking, but generally increases the accuracy of the model without the risk of lowering it.

2.3.2 Hyper-parameter Tuning

The main hyper-parameter we are tuning is the size of the bounding box.

While YOLO provides a very accurate bounding box that very tightly encapsulates the ball - we have found that there is a "sweet spot" for increasing the size such that it is better able to track the ball with all the hands dribbling it, with it going between people's legs, and going through nets.

While increasing the size of the bounding box may decrease the IOU score of frames where the ball is firmly tracked, it better guarantees that the tracker will actually track the ball and not get caught on some other object, thus increasing the overall IOU score of the video. That said, there is still an upper limit for the increase of the bounding box size. Using a box that it is too big generally causes the tracker to track the background of the ball in the initial detected frame or the player that is manipulating the ball.

2.3.3 GOTURN

Generic Object Tracking Using Regression Networks (GOTURN) is a single object deep learning based offline tracker.[2] While GOTURN is capable of tracking at 100 fps, this required the use of high-end graphics cards, such as the Titan X, that we did not have access to. GOTURN live tracking ran at less than 15 fps on the most powerful system we had access to. Using GOTURN would have forced us to endure long wait times for training and testing that we did not have the privilege of enjoying. Additionally, GOTURN was developed to track whole, generic objects. This meant that the tracker would often deviate from tracking the ball to tracking the player controlling the ball.

2.3.4 CSRT

The Channel and Spacial Reliability Tracker uses discriminative correlation filters to track objects in near real-time (CSR-DCF approach).[1] It uses a spatial reliability map to adjust the filter support to the part of the object suitable for tracking, which enlarges the search region and most importantly, improves the tracking of non-rectangular objects. The DCF uses ciruclar correlation to implement learning with a Fast Fourier Transform. The filter was trained on many examples that contain unrealistic, wrapped-around circularly-shifted versions of the target, perfect for the the tracking of spinning, blurry, high-velocity basketballs. [1] Combined with the fact that CSR-DCF is a model free approach (making it perfect for our weaker computer systems), it was quickly decided to be the main focal point of our project.

3. Results

In our research, we did not find any publicly available datasets that have localized basketballs in video. Instead, we opted to use more generalized datasets that have 'Sports Ball' (or similar) available as a class.

The main dataset we used for validation was the A2D (Actor-Action Dataset) from the University of Michigan. [5] This dataset contains not only basketballs, but a variety of localized objects. The dataset provides the input videos, localized objects in image form, and the raw matrices of the bounding boxes in MatLab. At a high level, we parsed the dataset to get only the videos containing a ball performing an action (for example rolling or flying) and then parsed the compiled MatLab to gain access to the raw bounding boxes of the balls in the video, and then proceeded to compute the IOU (intersection over union) score.

How we validated, parsed the MatLab, and got only the ball videos is discussed in more detail below.

3.1. Validation

While a lot of large datasets would be more-or-less plug and play with a CV codebase, that was not the case with A2D as a sizeable portion of our effort went into integrating the dataset. There were a few reasons for this.

Firstly, the dataset didn't only contain videos of balls - it contained a variety of different actor classes performing a variety of different actions. We specifically only wanted videos where there were localized 'Sports Ball' frames. We accomplished this by parsing A2D's provided csv using Pandas and extracting the paths of only ball videos.

Secondly, the dataset didn't just provide the bounding boxes in a csv - this was a harder problem to solve than the first. Instead, A2D provides bounding boxes in compiled, not raw, MatLab files. This means we could not just open the MatLab files ourselves. Instead we had to use h5py to de-compile the MatLab files, then reverse engineer their structure, and then iterate over the bytes to figure out which bounding boxes were that of a ball.

Also, there were multiple MatLab files per video, each representing a frame and the localized objects in said frame. We had to glob all the MatLab files in the video directory and extract their name into a map to use them to cross-validate

After those two things were done, we could programmatically iterate over the videos in the dataset and evaluate our tracking system.

3.2. Evaluation

3.2.1 A2D IOU

To evaluate our tracking system, we look at all localized ball frames in the A2D video at hand. On each frame we compute the IOU (intersection over union) score with our estimated bounding boxes. We take the average IOU over the course of the video as one evaluation metric. We then provide a graph of the IOU score per frame.

3.2.2 YOLO IOU

Another way we evaluate performance in the absence of a labeled dataset is via this YOLO IOU method. This method is similar to the A2D method in which we are making a prediction bounding box and comparing it to a source of truth via IOU. But here, we have no dataset as a source of truth - instead we track the ball using the specified tracking algorithm (for example CSRT) and then we run YOLO on every frame to re-localize the ball and use the result of YOLO as the source of truth due to its high accuracy.

We feel that the A2D evaluation method is still superior but this is a method we came up with that still evaluates decently well and does not require a dataset.

Figure 1. Example of YOLO truth box in green. CSRT tracked box in blue

3.2.3 Precision and Recall

Precision and recall are valuable metrics that allow us to glean more insight on the model's performance. Precision is calculated as the number of true positive predictions divided by total positive predictions. Recall is defined as the number of true positive predictions divided by the sum of the number of false negatives and the number of true positives.

For each input video, precision and recall were calculated based off of the number of available ground truth frames. For the YOLO detector-based evaluation, these were the frames where YOLO could detect a sports ball. For the A2D dataset, precision and recall were evaluated based off of the three pre-labeled frames of each video.

3.2.4 Code

When we export the localized video file, we will also draw the A2D source of truth so the user can get visual feedback

on where their computed bounding box was vs the source of truth bounding box.

Finally, with the *–a2d* flag the user can run the program against all of A2D and get the metrics for all videos should they want a very concrete and thorough analysis. The user can also specify the number of A2D videos they want to evaluate against should they not want to run on all of them with the *–a2d_num* flag.

3.3. Contour Network

The contour network definitely performed the worst out of our 3 methods. More often than not, the detected blob would be the basketball, as it was consistently the highest point blob unless the ball was in the air. As such, it could only track balls that were flying high in the air (such as an attempted shot at the basket) while completely failing on dribbling and below-the-shoulder passes. Specific, quantified results were not gathered for this approach, as the approach was abandoned long before validation and evaluation metrics for this problem were written.

3.4. Overlap Tracking

The simple overlap based approach did not perform nearly as well as expected. Often times, the YOLO detector was unable to find the ball if there was anything remotely close to obstructing it. This is clearly demonstrated in 3 consecutive frames displayed in Figure 2, where YOLO is only able to detect the ball in the first frame. In the second frame, the ball is slightly obstructed by the corner of the jumbotron above it and is noticeably more blurry. In the third frame, the ball is completely obstructed by the backboard and is unable to be detected. Because of the extremely long runtime of this approach (needing approximately 1 second per frame), it was not tested on the A2D dataset. Overall, it is clear that simple overlap based tracking is unsuitable for basketball, where the ball is easily hidden behind players, obstructed by hands, or moving too fast for a model that only looks at a single frame in isolation.

Figure 2. Overlap Tracking

3.5. YOLO + Tracking Algorithms

We would say that in most cases, YOLO in combination with another tracking algorithm was the most performant solution. Both algorithms were able to keep tracking the ball even when it was obstructed by other objects, such as the backboard, or when it was out of frame. The CSRT tracker, however, was able to track the ball specifically much better than GOTURN, as demonstrated in Figure 3 and 4.

We thus decided to run the CSRT tracker across 70 samples in the A2D dataset, the results of which are shown in Figure 5. IOU for a sample peaked at slightly less than 0.7. Precision varied dramatically, ranging from 0 to 1.0. Recall for a sample was either 0 or 1. This meant that the model either completely failed to track the ball on the evaluation frames, or tracked it without predicting any false negatives.

4. Discussion

To briefly touch on how we would improve our project, one of the key things we think would benefit us is having YOLO trained just on localized basketballs. We think this would lead to significant performance gains as basketballs appear harder to detect than the likes of a soccer ball, for example. A soccer ball is a black and white checkered object on a green platform that moves in parabolic and predicable ways - we saw that when we ran our detector on soccer datasets specifically the performance was very good.

Let's compare this to a basketball, its a beige - almost skin-tone - colored object, being dribbled by a human (who has skin-tone colored skin) on a beige basketball court. Ad-

Figure 3. YOLO + CSRT Tracking

Figure 4. YOLO + GOTURN Tracking

ditionally the basketball does not move in predicable ways as one of the main tenets of the sport is being able to get around your opponent by manipulating the movement of the ball. Because of these unique features of basketball, if we had access to a large dataset of localized basketballs training our YOLO output on that is something we think could lead to improvements for future work.

5. Conclusion

Overall, the combination of the YOLO detector with the CSRT tracker proved to be a successful approach to detecting and tracking balls throughout video in a processing power limited environment. It performed better than an overlap variant, which greatly suffered due to being unable to detect balls at high speeds and behind obstructions. It also performed better than the GOTURN tracker, which suffered from being unable to isolate the ball from surrounding, larger objects. The combination of YOLO and CSRT proved to be very complementary; the backwards tracking allowed the CSRT tracker to makeup for the weaknesses in the YOLO model when it failed to detect balls in initial frames.

References

[1] Luka Cehovin Zajc Jir'i Matas Alan Lukezic, Tom'as Voj'ir and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. 2018. 2

[2] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. 2016. 2

[3] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 1

[4] Constantin Toporov. Ball tracking in volleyball with opencv and tensorflow. 2020. 1

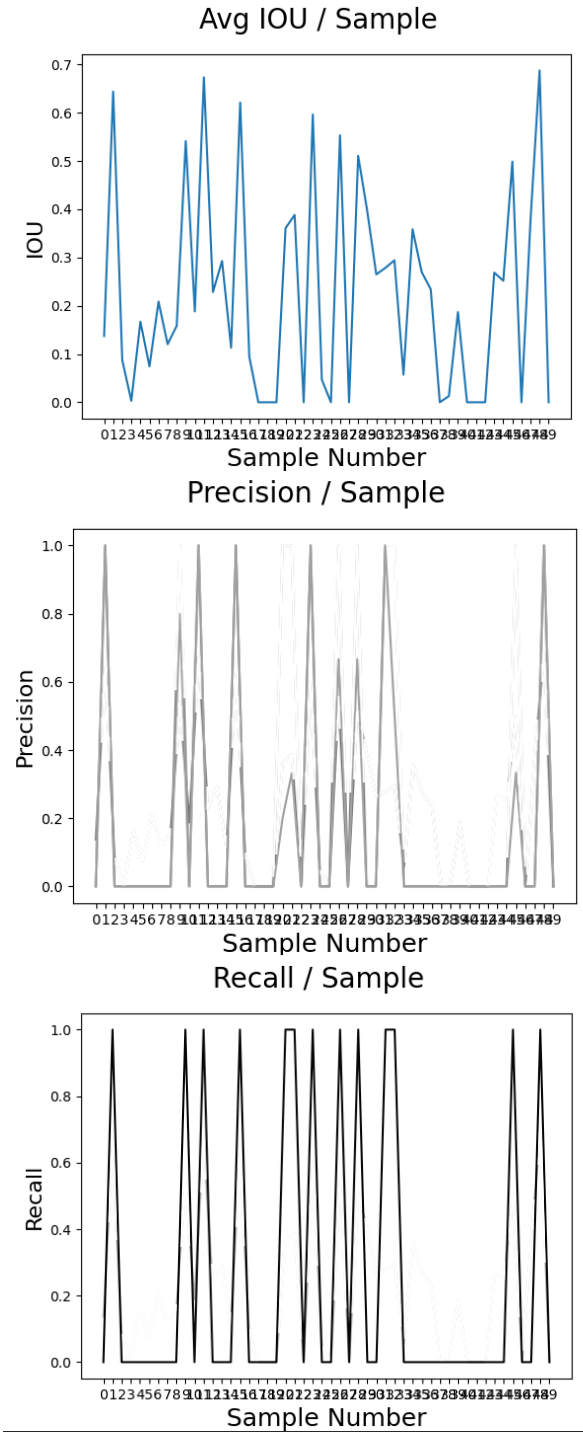

Figure 5. YOLO + CSRT on A2D

[5] Chenliang Xu and Jason J. Corso. A2d: A dataset and benchmark for action recognition and segmentation with multiple classes of actors. 2015. http://web.eecs.umich.edu/~jjcorso/r/a2d. 3