Final project proposal

Marine Mercier
MVA
marine.mer98@gmail.com

Abstract

Ball tracking in volley ball games can enrich the viewing experience and give useful insights concerning players performance and game strategy. The problem of ball detection and tracking is very challenging because of the wide variation in the appearance of the ball over frames. The ball can be close to the camera or far and small, hidden or distorted due to high speed. To overcome these challenges, I develop a two-step algorithm, first I generate a set of ball candidates on each frame, then I use trajectory information to locate the ball.

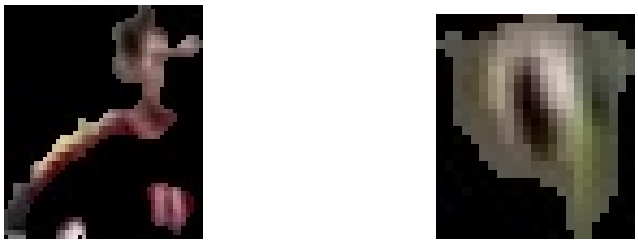
Introduction

The detection step is very important, indeed it is impossible to track the ball if it is not correctly detected. Many techniques for ball tracking are based on color, but the color of the ball can change in Volley (not at professional level though), so these techniques are useless. The first naive detection method I tried is based on motion. The ball is selected among moving object, either as the highest moving object or with a blob model trained on moving objects. Then I trained YOLOv5, pre-trained on the COCO dataset, with volley images.
The tracking can improve detection, as we can use ball trajectory to guess the ball position on missing frames. The Kalman Filter produces estimates of hidden variables based on inaccurate and uncertain measurements, and is therefore widely used in object tracking. Then I used the approach proposed in [1] to connect the track-segments proposed by Kalman filter and find the global trajectory of the ball.

1. Datasets

I mainly used 6 videos from professionals in the Austrian Volley League [8]. These videos are annotated with volleyball activity classes, but the ball is not annotated. The videos are taken in the same gymnasium, with different lightnings. The orientation of the camera is roughly the same on all videos.

A dataset composed of blobs extracted from these videos was created by [7], and I used it to train the blob model. Nevertheless, the images are very small, in moreover some images are wrongly labelled.



(a) Not a ball (b) Ball

Figure 1: Blob dataset examples

To train the YOLO model on proper ball images, I hand-labelled myself 650 images from 5 videos (4 hours work). I used roboflow to create the COCO annotation file. The main difficulty is the image extraction. In high level men's volleyball, the points are often very short and the breaks between two points are long. Getting enough interesting images from game phases was particularly time consuming. I didn't resized the images (1920*1088) as the ball can be really small in the background. In moreover, some images contain more than one ball (balls on the side of the field, or play during warm-up phases).

Last but not least, I recorded one of my own training to test the model in a new environment.

2. Detection

2.1. Motion detection

These methods based on motion detection were proposed by [6], and implemented in [7].

First, a set of moving objects is created for each image on the video. OpenCV tools detect moving object and remove the background. Then I filtered these moving objects according to their size and shape (the height and width ratio must be around one, the blob must contain a minimum number of pixels...).

The first naive method uses a rule saying that the ball is the highest moving object. Nevertheless, this method cannot track the ball as soon as it passes under the net, which is already a problem. In moreover this rule is not always true, depending on the angle of the camera, the presence of spectators, a referee, some objects can be higher than the ball.

Then I trained a model to identify if a moving object is a ball or not. This problem is a classic binary classification task. Given that blob images are very small, too much convolution layer would lead to over-fitting. VGG16 is a convolutional neural network model for classification[5], composed of convolution layers with the rectification (ReLU) non-linearity, combined with maxpooling then dense layers with ReLU and the last layer is a softmax layer. The simplified model proposed by [6] has two convolutional and maxpool layers, and two dense layers.

Nevertheless, the model performs badly. 72% of the ball detected are false positives, while 16% of the actual balls are missed (tested on 200 images). There are two type of false positives, some appears in random places and random times, some are due to the model consistently mistaking an object for a ball. Detections of the model appear in red on Figure 2, we can see an image with lots of false objects detected under the net.

Figure 2: Kalman Filter with the blob-model. *Red circles are ball detections, lines are segments tracked by Kahlman Filter based on previous detections.*

2.2. YOLOv5

Region proposal classification networks perform detection on different regions of the image. YOLO[2] (You Only Look Once) simultaneously predicts multiple bounding boxes and class probabilities for those boxes, it directly optimizes detection performance, and can be used in real time detection. The imput image is divided in a SxS grid, if the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. YOLOv5 is pretrained on the COCO dataset, which includes a 'sports ball' category. I used transfer learning in order to train the model on new ball images while keeping the knowledge gained on COCO.

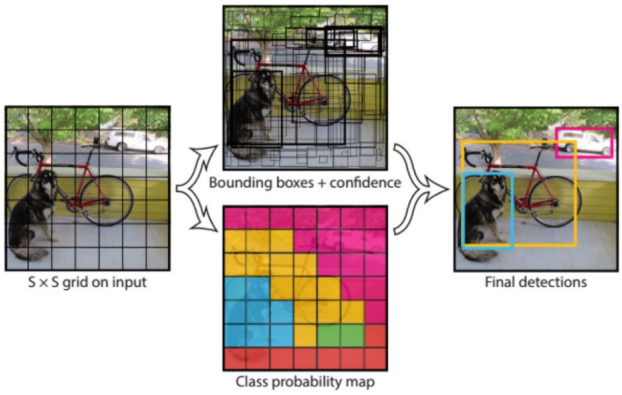


Figure 3: YOLO model[2]

After training, YOLOv5 produces 0% of false positives and 8.6% of false negatives (tested on 200 images). I used the smallest version of the model, YOLOv5s. YOLOv5 performs much better than the previous detection model. In moreover, the model can probably be improved if trained on more images. Indeed, it is recommended to train YOLOv5 on more than 1500 images, but I only labelled 550. As we can see on figure 3, the ball is not detected when it passes the white panel in the background. But dataset contains no images of the ball in front of this pannel, if the model sees more similar images, it will probably be able to detect the ball in these conditions.

3. Tracking

The trajectory process I developped consists of two steps. I first connect neighboring ball candidates in adjacent frames to form trajectory segments with Kalman Filter. Then I construct complete trajectories with these segments. [1] proposes a similar method, with an additional step which uses physics of the ball to validate trajectories or not.

Figure 4: Kalman Filter with YOLO. *Red circles are ball detections, lines are segments tracked by Kahlman Filter based on previous detections.*

3.1. Kalman Filter

The Kalman Filter[3] is a recursive algorithm, which uses a series of measurements observed over time, in order to estimates unknown variables of the ball trajectory. This filter has a prediction phase and a correction phase. During the prediction phase, it estimates the current state variables which is the position of the next ball. Then, if the prediction of the Kalman filter is close enough to a ball detection, this ball detection is added to the trajectory and the filter is updated and corrected. This filter is widely used in multi-objects tracking, including in sports [4].

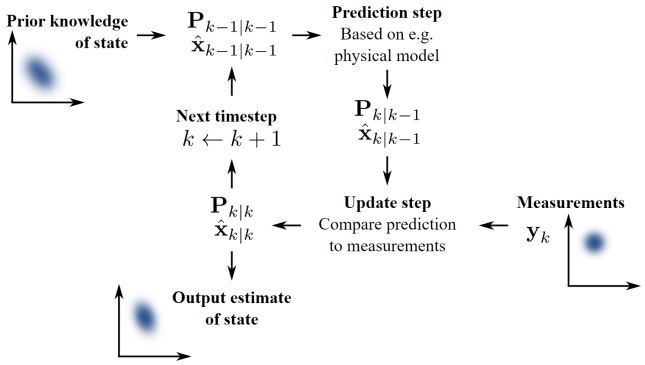


Figure 5: Kalman filter operation

In order to improve the performances of the Kalman Filter, I added some physics insights to the model. The ball is subject to the gravitational force toward the ground. With this information, the filter makes better predictions, therefore it is easier to track the ball. As you can see on figure 2, false predictions of the blob model makes it hard to track the ball under the net. The filter creates many false trajectories, either static or zigzagging among the false detections. Therefore, I added a filter, which limits the fast changes of direction. The segments produced by the Kalman filter with the blob model detections are visible in figure 2, and those produced with the YOLO detections are visible in figure 4.

3.2. Construction of complete trajectories

The Kalman Filter only creates segments of trajectory. If the ball disappear or is not detected for more than 5 frames, the filter can't predict the new position of the ball with accuracy. Therefore I have to connect these segments in order to have the full trajectory.

The generation of the full trajectory can be done in three steps described by [1]. During the Find stage, for each segment T_i, find other segments T_j that begin when T_i end, and whose first detection is close to the last track of T_i. Then, if T_i is longer, use a polynomial to fit T_i and estimate the ball position on the frame when T_j begins. If the distance between the first point of T_j and T_i's prediction is smaller than a threshold, connect the two segments, this is the connect stage. Last but not least, the extend stage is used to estimate the ball position on frames where the ball was not detected.

4. Results

4.1. With Blob Model

As we expected, generating trajectories with the blob detection model is complicated. When the ball is above the net, the trajectory is well extracted, but when the ball goes under the net, the algorithm might connect the trajectory to a wrong segment, loosing track of the ball.

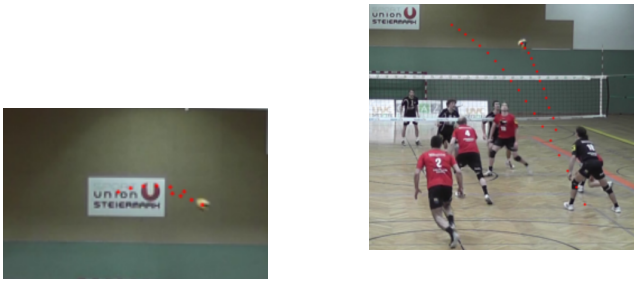
4.2. With YOLO Model

On the other hand, the trajectory extraction works really well with the YOLO model. We can easily follow the position of the ball over frames (figure 6). The segments are well connected, and the estimations of the ball position on missing frames are pretty accurate. With the tracking, the ball position is correctly detected in 92% of the images (tested on 200 images), which means that the tracking improves ball detection.The estimation of missing ball can lead to slight errors in trajectory (figure 7).

4.3. Test on a new environment

I tested my algorithm on volley-ball videos in another place, with an other angle and other players. The YOLO detection algorithm doesn't perform as well as on the previous dataset [8]. However, the model has only been

trained on that dataset, so it is normal that it does not generalize very well. By building a larger dataset of ball images, the model should work better. We can still track the right ball among the false detections (figure 8).



(a) Balls overlap

(b) The ball did not bounce off the player's arms

Figure 6: Wrong trajectory calculation

Figure 7: Successful tracking

Figure 8: Tracking in a new environment

References

[1] Wei-Ta Chu, Chia-Wei Wang, and Ja-Ling Wu. Extraction of baseball trajectory and physics-based validation for single-view baseball video sequences. volume 2006, pages 1813–1816, 07 2006. 1, 2, 3

[2] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016. 2

[3] Kenshi Saho. Kalman filter for moving object tracking: Performance analysis and filter design, kalman filters - theory for advanced applications. IntechOpen, 10 2017. 3

[4] Yongduek Seo, Sunghoon Choi, Hyunwoo Kim, and Ki-Sang Hong. Where are the ball and players? soccer game analysis with color-based tracking and image mosaick. In Alberto Del Bimbo, editor, *Image Analysis and Processing*, pages 196–203, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 3

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 2

[6] Constantin Toporov. Ball tracking in volleyball with opencv and tensorflow. 2020. Computer vision and neural networks in SportTech. 1, 2

[7] tprlab. Github repository vball. 2020. https://github.com/tprlab/vball. 1

[8] Georg Waltner, Thomas Mauthner, and Horst Bischof. Improved Sport Activity Recognition using Spatio-temporal Context. In *Proc. DVS-Conference on Computer Science in Sport (DVS/GSSS)*, 2014. 1, 3