Trajectory Prediction using Kalman Filter with OpenCV

Jacopo Clocchiatti
Matricola: 229701
Email: jacopo.clocchiatti@studenti.unitn.it

1. Introduction

In this era where computer vision is being applied in multiple fields, sports are one of the most suitable field to leverage the possibilities given by the innovation in computer vision. Many sports can make use of computer vision for different purposes, from analytical purposes to improving the performances of the athletes, e.g. leverage pose estimation algorithms for golfers to improve their drive movement or to analyse the shooting motion of a basketball player.

Basketball is indeed the sport of choice of this project but it has a more analytical purpose. We developed a model to automatically predict the trajectory of a shot in a free throw situation. We improve already existing project that need to manually input the area to track, and then predict, with the addition of a deep-learning object detector model, YOLO, to automatically find and track the basketball.

2. Related Works

There are many works that leverage the use of Kalman filter for prediction intents in a numerous fields. With regards of basketball specifically Kalman filter has been used for various purposes, from estimating the outcomes of matches [1] [3] to self training and posture estimation [2]. Similar works to our projects were carried out by [8] [9] but their works needed a manual input for the area to track. We start from their developments and improve them adding an automatic detector for the basketball and multiple tracking algorithms.

3. Solution

The proposed solution consists in using a YOLOv8 object detector to first find the basketball and then either utilize the same detector or use Camshift algorithm to track the basketball in the subsequent frames. For each frame a Kalman Filter is used to make the trajectory prediction.

3.1. YOLOv8

YOLOv8 [5] is the newest state-of-the-art YOLO model developed by Ultralytics, who already developed the fifth version of the YOLO series. It is built as a unified framework for training object detection, instance segmentation and image classification models. There are five models in each category of YOLOv8 models for detection, segmentation and classification, from Nano to Extra Large. YOLOv8 models seems to perform better compared to previous YOLO models. YOLOv8 is an anchor-free model, i.e. it predicts dorectly the center of an object instead of the offset from a known anchor box. YOLOv8 it also use mosaic augmentation for the training routine, but not for the whole training as it degrade performance, so it is used mainly for the first part of training.

3.2. Kalman Filter

The Kalman filtering [11] is relatively simple state-space algorithm to produce estimates of the hidden variables based on uncertain and inaccurate measurements. It predicts the systems future state based on past estimation. It can find the best estimate from noisy input data by filtering out the noise. The initialization of the filter has two inputs: the initial state of the system and the uncertainty of that initial state, they do not need to be precise because the algorithm eventually converges. After the initialization the Kalman filter consists of two steps executed in a loop:

1) Prediction step, where it predicts the next state of the system form previous measurements
2) Update step, when it estimates the current state from the measurements provided

The initialization is performed only once for the first measurement. Once initialized, the Kalman Filter will predict the system state at the next time step and it provides the uncertainty for this prediction. Once the measurement is received the Kalman Filter corrects the current state's prediction and uncertainty.

3.3. Camshift Algorithm

The Camshift (Continuously Adaptive Mean Shift) algorithm [6] is a colour-based object tracking method introduced by Gary Bradski to reduce computational complexity and to deal with problems such as image noise, distractors, irregular object motion due to perspective and lighting variations. The Camshift algorithm derived from the mean

TABLE 1. TRAINING RESULTS AFTER 100 EPOCHS FOR EACH MODEL SIZE

Type	Size(MB)	Batch Size	Training Time	Inference Time per Image (ms)	$mAP50_{train}$	$mAP50\text{-}95_{train}$	$mAP50_{valid}$	$mAP50\text{-}95_{valid}$
Nano	6.2	79	2:01:04.800	7.6	0.959	0.623	0.958	0.623
Small	22.5	42	2:07:01.200	12.9	0.959	0.626	0.956	0.626
Medium	52.0	23	2:49:48.000	16.0	0.958	0.606	0.952	0.605

shift algorithm which is responsible for finding the center of the probability distribution of the object to track. As the meanshift needs, the Camshift also needs to first choose the initial location of the search window. As it is an algorithm based on colour it has some limitation. One of the major ones is that it cannot track the desired target when the background is of the same colour.

4. Experiments

4.1. Dataset

There are not many dataset available that are build with the aim of detecting basketball. There are only some images that include basketballs in the bigger dataset like COCO, but they are rare. In the end a dataset was built using images of interests extracted from other datasets available online. Those images were then imported in Roboflow platform [4] where they were manually tagged and labeled, after the images were splitted in the three usual sets (train, validation and test).

The final dataset consist of 3269 total images, splitted in 2224 images for the training set (68%), 318 images for the validation set (10%) and 727 images for the test set (22%).

4.2. Training

YOLOv8 models were trained in Google Colab using the available GPU (Nvidia Tesla T4 16GB). The models have been fine-tuned starting from the given pre-trained ones. Nano, small and medium models were chosen to get enough accuracy to detect the basketball while having near real-time detection speed. The three models were trained using the same dataset with the images resized to 640 for each dimension, the same split are used but with automatic batch sizes, to better use the GPU's memory, and have been trained for 100 epochs also using early stopping to avoid overfitting. AdamW optimiser [12] is being used and also data augmentation is being applied on training data. More specifically the operations that are being applied are blur, conversion to grayscale and histogram equalisation (i.e. Constrasted Limited Adaptive Histogram Equalisation [13]). As the Table 1, we see that for similar performance results the nano version of YOLOv8 offer the best trade-off between performance and speed. Note that the small version and especially the medium one can be trained for more epochs before they overfit, for this project they were trained for the same amount of epoch for comparison purposes.

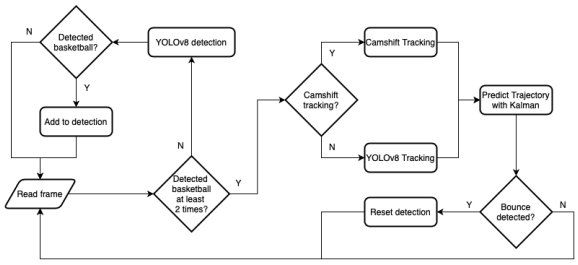

Figure 1. Flowchart of the final system

4.3. Final System

A simplified diagram of how the final system works is shown in Figure 1.

The final system read a video stream (currently only for videos, but it can be easily extended to live video stream like webcam feeds), and for each frame it first try to detect the basketball using the YOLOv8 model of choice. Once the basketball was detected (two times for the first prediction), the system switch to the tracking mode, where the user can decide to use the Camshift algorithm (best for situation where the gym walls and the basketball colours are significantly different) or continue to use YOLO. In the tracking phase, the system track, for each frame, the basketball and predicts its trajectory using Kalman filter and displaying the prediction. The prediction displays the probable positions of the basketball in the rest of its trajectory. Towards the end of the predicted trajectory the uncertainty is much bigger than in the earlier parts so the system takes into account this uncertainty, i.e. the position displayed is much bigger than the basketball dimension to better explain this position uncertainty. When a bounce is detected, the system reset and goes back in "detect-mode" to find the basketball and then turn back tracking it.

5. Open Issues

The proposed system works in the expected way but it has still some room for improvement.

The main issue is that in a normal scenario the tracking system that use only YOLO model can't give a real-time track, this can be solved using devices that are equipped with GPU or to make use of sparsification tools like Neural Magic® [7]. Another issue is that even the bigger model that is used in this work can be improved in terms of accuracy, especially in challenging situation like when the basketball court has colours similar to the basketball.

There are two ways that can address this problem, and they can be used together for a better result. The first option is to use a bigger/deeper model, like the large version of YOLOv8, this, given the same dataset, can bring minimal but effective improvements in the detection. The second option is more data-centric, that is increase the size of the dataset adding more images of basketball, preferably in match situations so as to have training data that reflect more the situation in which the model will be used.

6. Conclusion

We developed a system for automatic tracking and trajectory prediction of free throws. We improved on existing methods with the addition of a deep learning object detection model to find the basketball in the scene. The performance are good in controlled situation, for which the system is developed for.

Future improvements can be made, in addition to the already mentioned ways to solve the present issues. A detector for the rim can be added to automatically check if the shot was successful or not. Also a pose estimator can be inserted in the system to detect the shot pose. In this way an analytics system can be build where the user can leverage all these information to see what works and what not in its shooting form to improve its shooting ability.

References

[1] Poropudas Jirka, Kalman filter algorithm for rating and prediction in basketball, Helsingfors universitet, 2011

[2] Egi, Yunus. "Basketball self training shooting posture recognition and trajectory estimation using computer vision and Kalman filter" Journal of Electrical Engineering, vol.73, no.1, 2022, pp.19-27. https://doi.org/10.2478/jee-2022-0003

[3] Manner, Hans. "Modeling and forecasting the outcomes of NBA basketball games" Journal of Quantitative Analysis in Sports, vol. 12, no. 1, 2016, pp. 31-41. https://doi.org/10.1515/jqas-2015-0088

[4] Dwyer, B., Nelson, J. (2022), Solawetz, J., et. al. Roboflow (Version 1.0) [Software]. Available from https://roboflow.com. computer vision.

[5] Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

[6] Bradski, Gary R.. "Computer Vision Face Tracking For Use in a Perceptual User Interface." (1998).

[7] https://neuralmagic.com

[8] Pérez Pablo Saura, Garcìa Alberto Ruiz, "Anàlisis de eventos deportivos mediante visìn artificial", Universidad de Murcia, 2017.

[9] https://github.com/khw11044/kalmanFilter

[10] https://arshren.medium.com/an-easy-explanation-of-kalman-filter-ec2ccb759c46

[11] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. Transaction of the ASME - Journal of Basic Engineering, 35-45

[12] Loshchilov, Ilya, and Frank Hutter. "Decoupled weight decay regularization." arXiv preprint arXiv:1711.05101 (2017).

[13] S. M. Pizer, R. E. Johnston, J. P. Ericksen, B. C. Yankaskas and K. E. Muller, "Contrast-limited adaptive histogram equalization: speed and effectiveness," [1990] Proceedings of the First Conference on Visualization in Biomedical Computing, Atlanta, GA, USA, 1990, pp. 337-345, doi: 10.1109/VBC.1990.109340.