Take home assessment report

Yongyu Deng

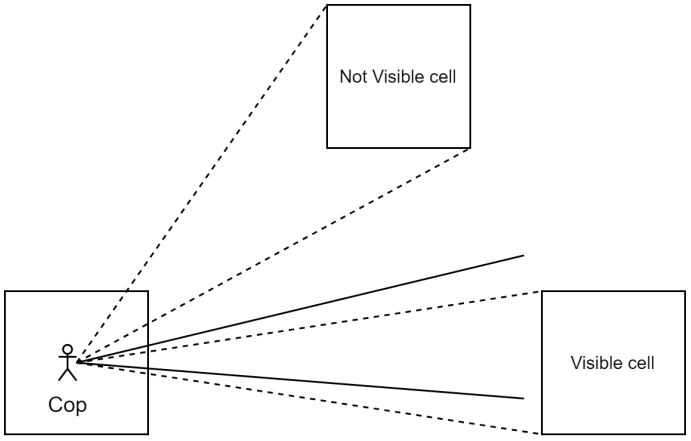
Task 1

Intuition

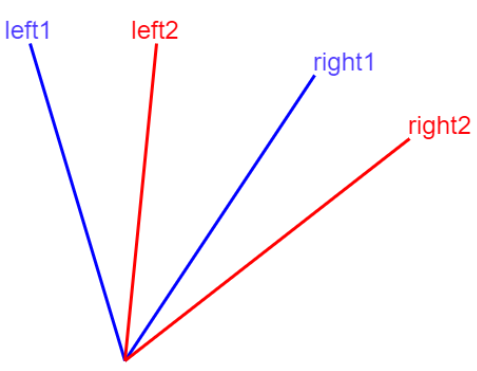
For each police, iterate through cells in the grid and check if the cell is visible to the police.

Algorithm

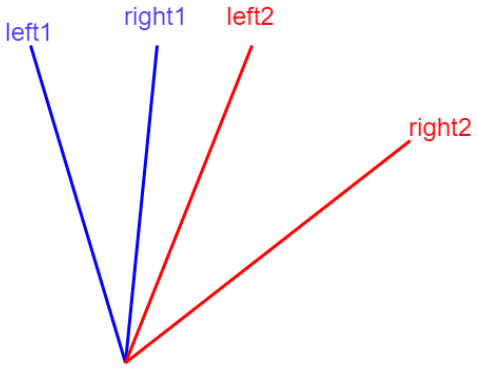
The key of the algorithm is to find how to determine if a cell is visible to a police given the FoV and orientation.



As the picture above shows, if the cop's FoV overlaps with the cell's FoV (define this as the minimum FoV for the cop to see the whole cell), then the cell is visible to the cop, otherwise not visible. To check FoV overlapping, we just need to check if **the rightmost of 2 FoVs' left sights** is on the left of **the leftmost of 2 FoVs' right sights**, as shown in the picture below.



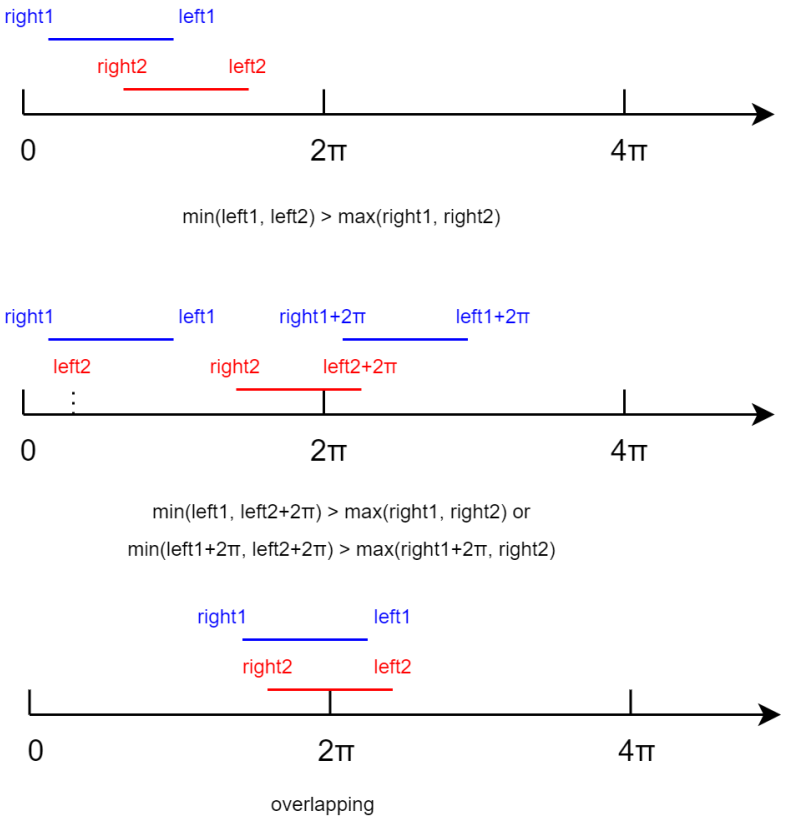
left2 is on the left of right1, overlapping



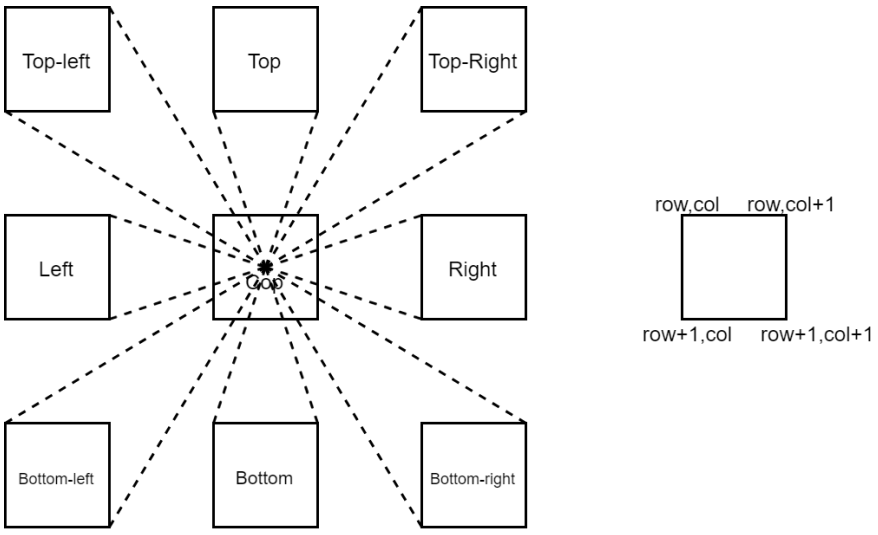
left2 is on the right of right1, not overlapping

We can do this by comparing radians left1_r, right1_r, left2_r, right2_r (range [0, 2π)). One thing to notice is that since we have limited the range of radians [0, 2π), once we see one of FoVs' left radians is less than right radians, we need to add an offset

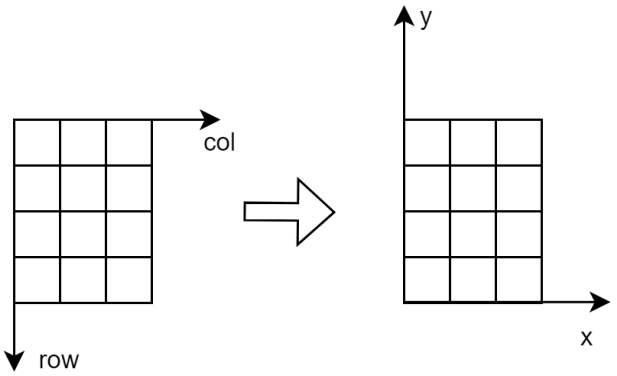
2π to left radians and compare in [0, 4π) space. If both FoVs' left radians less than right radians, that means both FoVs are crossing from [0, 2π) to [2π, 4π), they must overlap. Details are shown in the picture below.



To get the cell's FoV, here are the 8 conditions to find left and right radians.



To keep consistent with the given degree's coordinate system, we need to transform the row-col coordinates to x-y coordinates by (col, n_rows-row). The transformation process is given by the picture below. Then use atan2 to get the radians. Add 2π offset to negatives since atan2 output's range is ($-\pi$, π).



n_rows: number of rows

$$\begin{bmatrix} \cos(-90°) & -\sin(-90°) \\ \sin(-90°) & \cos(-90°) \end{bmatrix} \begin{bmatrix} row \\ col \end{bmatrix} + \begin{bmatrix} 0 \\ n\_row \end{bmatrix} = \begin{bmatrix} col \\ n\_rows-row \end{bmatrix}$$

The pseudocode is as following:

- Initialize empty list *cops_visible* and *seen* matrix with same size of grid and all 0.
- Iterate the grid and get positions of thief and cops.
- For every cop, check all the grid cells, if visible, set the cell in *seen* to 1; if see the thief, add the *cop_id* to *cops_visible* list.
- Iterate *seen* and find the closest cell that is not visible to cops.

Let C be the number of cops, n be the number of rows and m be the number of columns. The time complexity is O(Cmn), and space complexity is O(mn).

Results

Case 1
grid: [[0,0,0,0,0],['T',0,0,0,2],[0,0,0,0,0],[0,0,1,0,0],[0,0,0,0,0]]
orientations: [180,150]
fov: [60,60]

output: ([2], [2, 2])

Case 2
grid: [[0, 0, 0, 0], [0, 0, 0, 0], [0, 1, 'T', 0], [0, 0, 0, 0]]
orientations: [0]
fov: [30]

output: ([1], [1, 2])

Task 2

1. Promlem Analysis

This is an object tracking problem that has received significant attention in recent years as an active research direction. Object tracking is a challenging task due to several factors, including background clutter, occlusion, and illumination variation. In the given video, there are additional factors that further increase the difficulty of the task, including:

- Fast-moving small object: The soccer ball is a small object that moves at high speeds and changes direction frequently, making it a particularly challenging object to track compared to larger, more regularly moving objects like vehicles or people. The fast movement of the ball also causes motion blur in the image, as shown in Figure 1, further complicating tracking efforts. Additionally, small objects like the soccer ball may have low contrast with the background, as illustrated in Figure 2, which can make them difficult to detect and track.

Figure 1: Motion blur

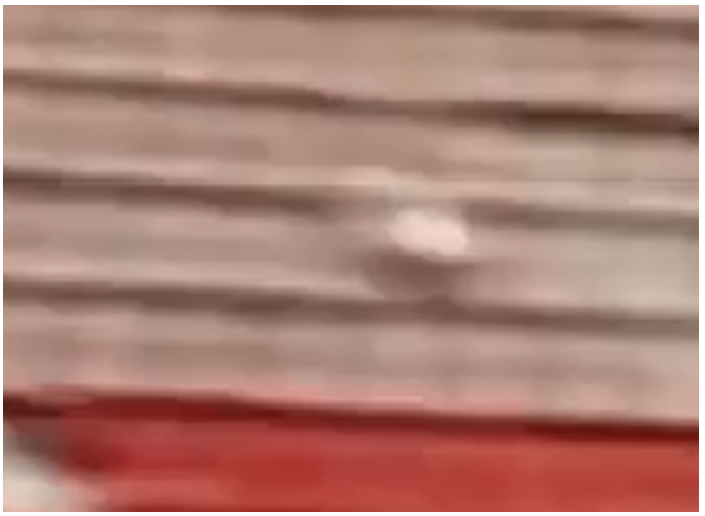


Figure 2: Low contrast with background

- Camera motion: The video was recorded by a walking person without a camera stabilizer, resulting in camera motion blur in the image. This camera motion blur makes it challenging to track the object accurately, as it becomes harder to predict and track the object's motion when the camera is also moving.

2. Experiments

2.1 Traditional Single Object Tracking Algorithms

At the outset of the project, I researched traditional single object tracking algorithms, many of which are readily available as pre-built algorithms in OpenCV for object tracking:

- BOOSTING: This is an older tracking algorithm that is based on the AdaBoost algorithm. The classifier is trained at runtime by learning on positive and negative examples of the object to be tracked. However, this tracker runs slowly (~20 fps) and is prone to losing track of the object when it moves too quickly.

- MIL: MIL is similar to the BOOSTING tracker, but with the addition of considering a small portion of the object's neighborhood as a positive example for the classifier. However, this tracker also runs slowly (~15 fps) and did not perform well for the given video.

- KCF: The Kernelized Correlation Filter algorithm is based on using correlation filters to estimate the position of an object in each frame of a video. This algorithm uses a kernel function to map image patches into a higher-dimensional feature space, where the correlation between the target object and candidate patches can be efficiently computed using FFT. This tracker is very time-efficient (~700 fps); however, it is unable to handle scale changes and fails to track the soccer ball at the beginning of the juggling.

- TLD: The Tracking-Learning-Detection (TLD) tracker combines tracking, learning, and detection to improve the accuracy and robustness of object tracking. The algorithm follows the object frame by frame and localizes its position learned from previous tracking, while simultaneously correcting the tracker if necessary. Although this tracker provides improved accuracy, it runs slowly (~10 fps) and is sensitive to changes in lighting conditions.

- MEDIANFLOW: This is a feature-based object tracker that computes the median of the motion vectors of the tracked object in a series of frames. While it is relatively fast and can track objects with moderate motion, it may struggle

to track objects with sudden and fast changes in motion, such as the soccer ball at the beginning of the juggling.

- GOTURN: This is a deep learning-based tracker that works by training a CNN to estimate the position of the tracked object in each frame of a video sequence. The tracker cannot handle occlusion well and loses track of the soccer ball at the beginning of juggling.

- MOSSE: The Minimum Output Sum of Squared Error tracker is a single object tracker that works by learning a correlation filter to track the target object in the Fourier domain. Similar to other feature-based trackers, MOSSE may struggle with handling occlusion and fast-moving objects. In the given video, it loses track of the soccer ball at the beginning of juggling due to its fast movement and motion blur.

- CSRT: The Channel and Spatial Reliability Tracking (CSRT) algorithm is a correlation-based tracker that combines spatial and channel reliability models to improve tracking performance. Among the single-object tracking algorithms tested, it achieved the best accuracy in the video and tracked successfully for the first half of the video. However, it was unable to track the soccer ball after it was occluded by the player.

2.2 SiamRPN++

SiamRPN++ was introduced in 2019. It builds upon the Siamese architecture for visual tracking, and incorporates a Region Proposal Network (RPN) to improve the accuracy of object localization.

SiamRPN++ uses a fully-convolutional Siamese network to compare the target object with candidate regions in the current frame, and generates a set of candidate proposals using an RPN. It then refines the proposals using a regression head, and finally assigns a score to each proposal based on how well it matches the target object.

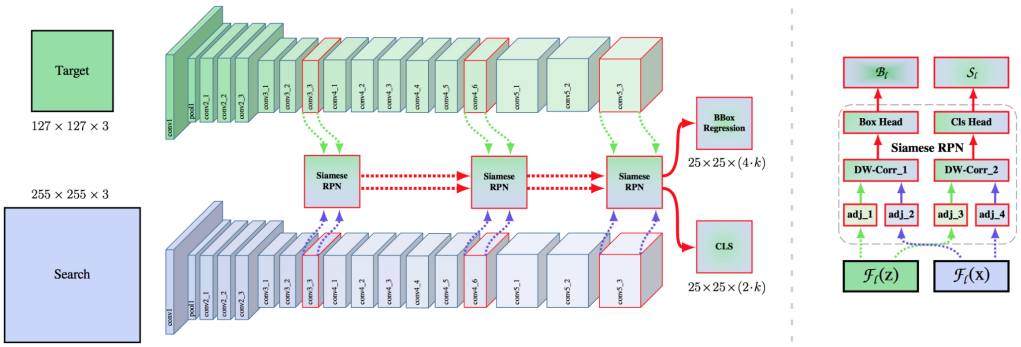


Figure 3: Given a target template and search region, the network outputs a dense prediction by fusion the outputs from multiple Siamese Region Proposal (SiamRPN) blocks. Each SiamRPN block is shown on the right.

In this task, I used MobileNetV2 as the backbone network because it's computationally efficient and lightweight, and still has relatively high accuracy. This algorithm works well for most parts of the video, however, it will fail to track the high speed ball after the player shoots as shown in Figure 5 and 6.

Figure 5: Before shooting

Figure 6: After shooting

2.3 Tracking by Detection

Tracking by detection is a tracking approach that involves detecting objects in each frame of a video sequence and then linking the detections across frames to form trajectories and it tends to be more accurate than SiamRPN++, especially in scenarios with occlusions or where objects undergo significant appearance changes over time. This is because tracking by detection relies on object detectors that are trained to handle complex scenes, while SiamRPN uses a simpler matching

approach that may struggle with more challenging scenarios. In this task, I used YOLOv8 as the object detection model.

2.3.1 Training Soccer Ball Detector

Firstly, I tried to use the pretrained model provided by YOLOv8 on our video, and I found that there's lots of false detection and detection failures, as Figure 7 showed.

Figure 7: False detection of YOLOv8 pretrained model

I think there 2 main reasons that the pretrained model will fail on this video:

- The YOLOv8 pretrained model was trained on the COCO dataset, and this is an unbalanced dataset. The number of class sports ball is much less than some majority classes in the dataset such as person and cars. Figure 8 and Figure 9 compare the number of classes in the COCO dataset's validation set.

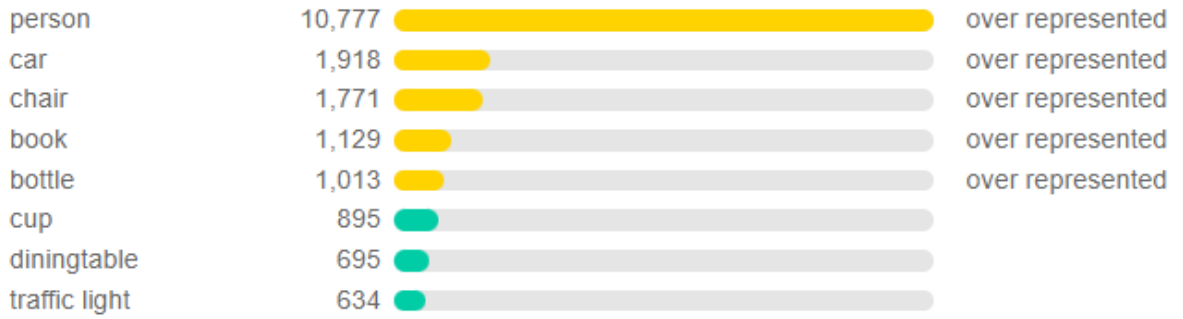


Figure 8: Majority classes are over represented

Figure 9: Sports ball is under represented

- Most of the sports ball images in the COCO dataset are static, which does not represent the scenario of our video.

Thus, I used a soccer ball dataset to train the ball detector. I have not found a dataset with fast moving soccer, so I manually annotated some frames of the video and trained the detector together to help the model to detect high speed balls. Because we have limited time and compute resources, and also high accuracy is not the highest priority, I trained the model from the YOLOv8 nano with 50 epochs.

Figure 10: Ball detector results

2.3.2 YOLOv8 with Simple Kalman Filter

I did a simple experiment using the YOLOv8 model with Kalman Filter and IoU matching. There are 6 states in the Kalman Filter: bounding box center (*x, y*), width and height (*w, h*) and velocity (*dx, dy*). It works very well on a surveillance video to track pedestrians, as shown in Figure 11.

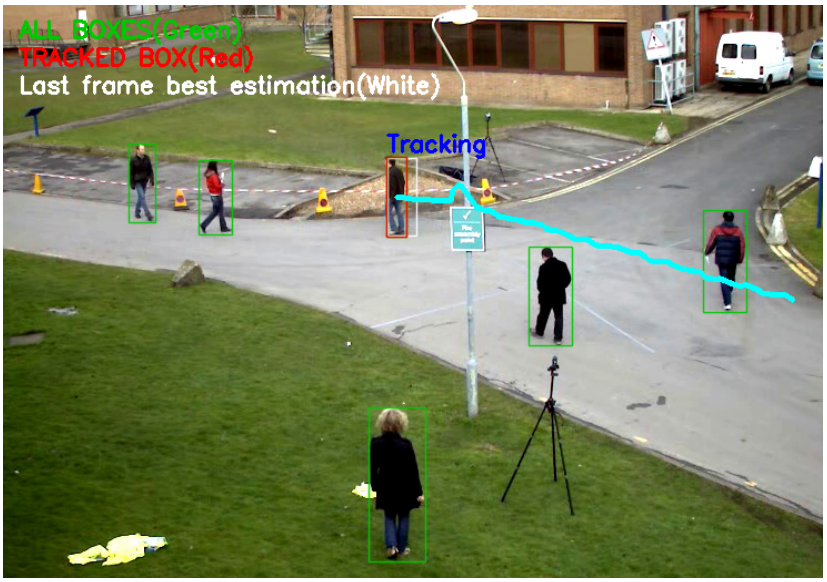


Figure 11: Tracking a pedestrian with simple Kalman filter

However, our video has a more complicated scenario. As mentioned before, the camera also has ego-motion, and the soccer ball's motion also has more variance in speed and direction, so it could be hard for the Kalman filter to predict motion with 6 states. When applying this method to our video, the tracker will lose track as shown in Figure 12.

Figure 12: Kalman filter failed to track

2.3.3 YOLOv8 with DeepSORT

DeepSORT is one of the most popular and widely used object tracking algorithms. The Kalman filter is also a very important component in deepSORT. Compared to the Kalman filter in the previous experiment, the Kalman filter in deepSORT has 8 states: bounding box center (*x,y*), aspect ratio *a*, height *h* and their respective velocities. The algorithm gets detections from our YOLOv8 model, Kalman filter tracking it and giving us missing tracks, compute Mahalanobis distance matrix and obtain appearance feature vector from feature extractor, and use Hungarian algorithm to solve association problem.

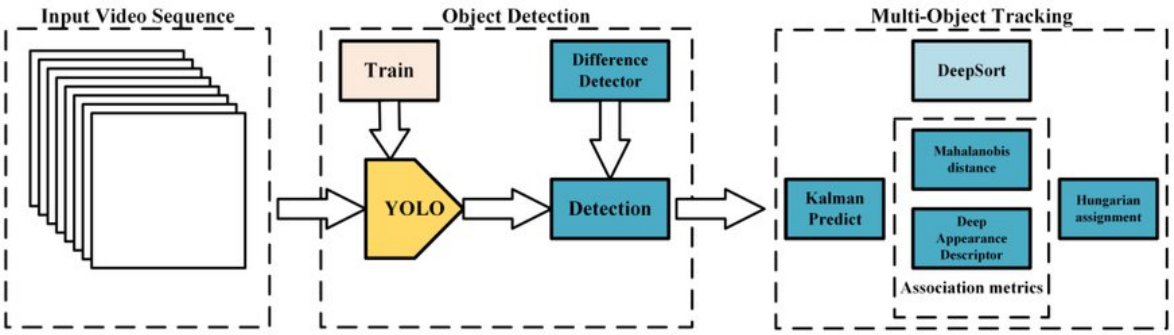


Figure 13: DeepSORT

In this task, I used the pretrained feature extractor provided by the paper. One problem of doing this is that the model is trained on MARS dataset, which is a large-scale person re-identification dataset, and suited for a people tracking context.

Despite this, the algorithm still works well for our video compared to some previous experiments, and it can successfully track some frames when the ball is flying fast. We can reasonably expect this algorithm will work much better once we have trained the soccer ball feature extractor on a large soccer ball dataset. Figure 14 shows 4 consecutive successful tracking of the shooting.

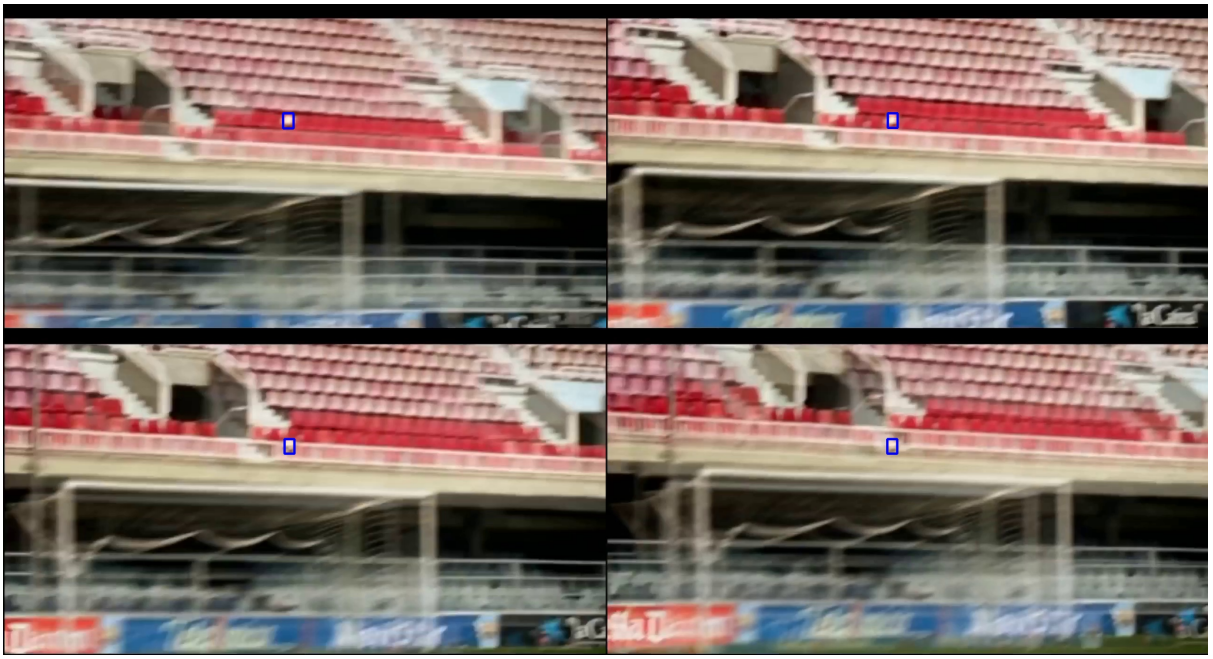


Figure 14: Successful tracking of flying soccer ball

3. Conclusion and Future Work

In this work, I studied several different methods in object tracking to solve this problem. Among these methods, SiamRPN++ and DeepSORT have relatively better tracking results. One benefit of using SiamRPN++ is that it's more time efficient than tracking by detection algorithms. For a scenario to track fast-moving small objects like this video, DeepSORT will have better precision.

With the limited time and compute resources, I only trained the soccer ball detector YOLOv8 nano version on a small dataset. More data and training need to be done in the future. Replacing the DeepSORT pretrained feature extractor by a specific soccer ball feature extractor will also be promising. One more thing I think is worth studying is that we can add a camera ego-motion estimation algorithm, and add them to the Kalman filter states in DeepSORT.

4. References

Wojke, Nicolai, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric." 2017 IEEE international conference on image processing (ICIP). IEEE, 2017.

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Li, Bo, et al. "High performance visual tracking with siamese region proposal network." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
Li, Bo, et al. "Siamrpn++: Evolution of siamese visual tracking with very deep networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Grabner, Helmut, Michael Grabner, and Horst Bischof. "Real-time tracking via on-line boosting." *Bmvc*. Vol. 1. No. 5. 2006.

Babenko, Boris, Ming-Hsuan Yang, and Serge Belongie. "Robust object tracking with online multiple instance learning." *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2010): 1619-1632.

Henriques, João F., et al. "High-speed tracking with kernelized correlation filters." *IEEE transactions on pattern analysis and machine intelligence* 37.3 (2014): 583-596.

Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." *IEEE transactions on pattern analysis and machine intelligence* 34.7 (2011): 1409-1422.

Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Forward-backward error: Automatic detection of tracking failures." *2010 20th international conference on pattern recognition*. IEEE, 2010.

Held, David, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer International Publishing, 2016.

Bolme, David S., et al. "Visual object tracking using adaptive correlation filters." *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010.

Lukezic, Alan, et al. "Discriminative correlation filter with channel and spatial reliability." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.