Rapport Machine Learning for visual Object Tracking

gautier.gally

december 2024

Contents

1 Single Object Tracking with Kalman (Centroid-Tracker)

1.1 Objectives

Develop a system capable of tracking a single object in a two-dimensional space by utilizing an existing object detection algorithm and integrating the Kalman Filter for smooth and accurate tracking.

1.2 Implementation

For each frame, the centroid of the object is detected using the provided object detection function. The Kalman Filter is then used to predict the position of the centroid for the current frame. Afterward, the predicted position is updated using the detected centroid to refine tracking accuracy. The visualization step displays the detected centroid, the predicted position, and the trajectory of the tracked object on the video frames.

1.3 Result

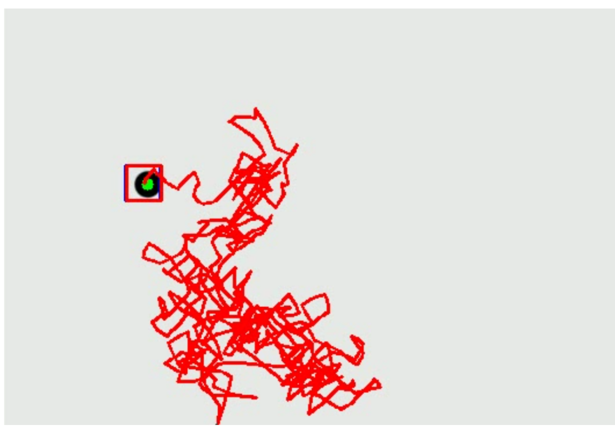

Figure 1: Output from tp1

The red square corresponds to the prediction of the object's position, while the blue bounding box represents the updated position after applying the Kalman Filter.

From the visualization, we observe that the blue rectangle successfully encapsulates the object, demonstrating that the algorithm functions as intended.

1.4 Challenges

Several challenges were encountered during the development of the tracking system. The first challenge involved ensuring the correct dimensions of the matrices in the Kalman Filter's update and predict methods. Any mismatch in matrix size caused computational errors that required careful debugging and validation. Another significant challenge was accurately displaying the object's detected centroid and predicted position on the video frames. Minor errors in coordinate calculations could create the illusion of algorithmic failure, significantly affecting visualization and overall tracking performance.

2 IOU-Based Tracking (Bounding-Box Tracker)

2.1 Objectives

The objective of this task is to develop a simple IOU-based tracker for object tracking and extend it for multiple object tracking (MOT). The tracker represents objects as bounding boxes and uses the intersection-over-union (IOU) metric to associate detections with tracks across frames. The implementation also involves processing pre-generated detections from a text file and managing the tracks effectively.

2.2 Implementation

The implementation begins by processing detection data stored in a MOT-challenge-like formatted text file. For each frame, the algorithm loads the detections, represented by bounding box coordinates and confidence scores. A similarity matrix is computed by calculating the IOU values between existing tracks and new detections.

The Hungarian algorithm is then applied to find the optimal assignment of the detections to the tracks. Depending on the assignment, tracks are updated, unmatched detections are used to initialize new tracks, and unmatched tracks are removed if they exceed a predefined threshold of missed frames. The results are visualized by overlaying bounding boxes and track IDs on the video frames. Additionally, the tracking results are saved in a text file in a format similar to the ground truth data.

2.3 Result

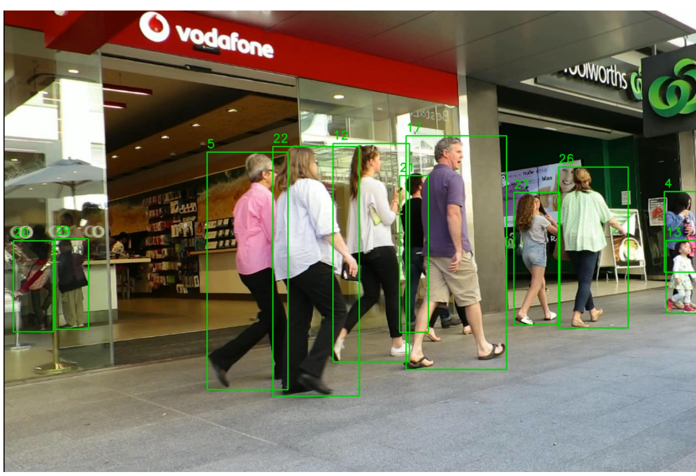

Figure 2: **TP2 result**

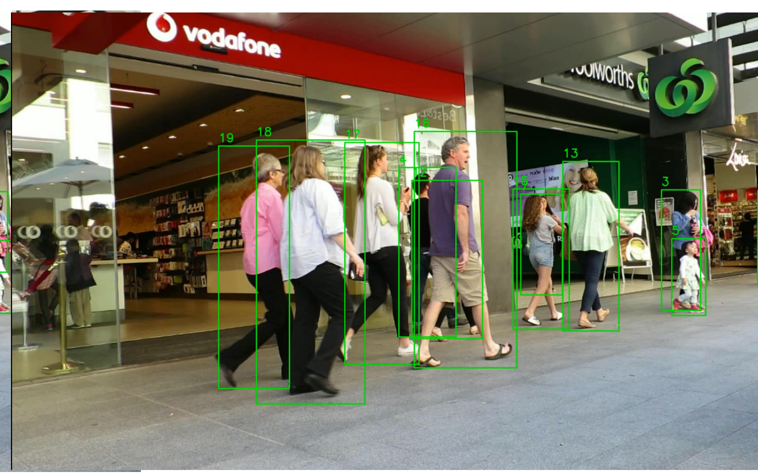

Figure 3: **Ground Truth**

We can see that my id is close to the id given in ground truth. However, some id like 5 have been taken twice by two different objects.

The system successfully tracks multiple objects in the provided video sequence. Bounding boxes and their associated track IDs are displayed, showing that the algorithm maintains continuity for each tracked object. The tracking results are saved in a text file, demonstrating compliance with the MOT-challenge format.

2.4 Challenges

During the development, a significant challenge was managing the dimensions of the similarity matrix and ensuring compatibility with the Hungarian algorithm. Another challenge involved maintaining track continuity when objects occlude or reappear in the frame, as mismatches could lead to tracking errors. Debugging and careful parameter tuning were essential to mitigate these issues and achieve reliable tracking results.
Handling the problem of sharing IDs: because we only use IoU sometimes, when an object disappears and another one appears in the same place, the ID of the disappeared object is assigned to the new object. I tried adjusting the maximum frames for which a detected object is saved, as well as adding a threshold to the best IoU match after applying the Hungarian algorithm.

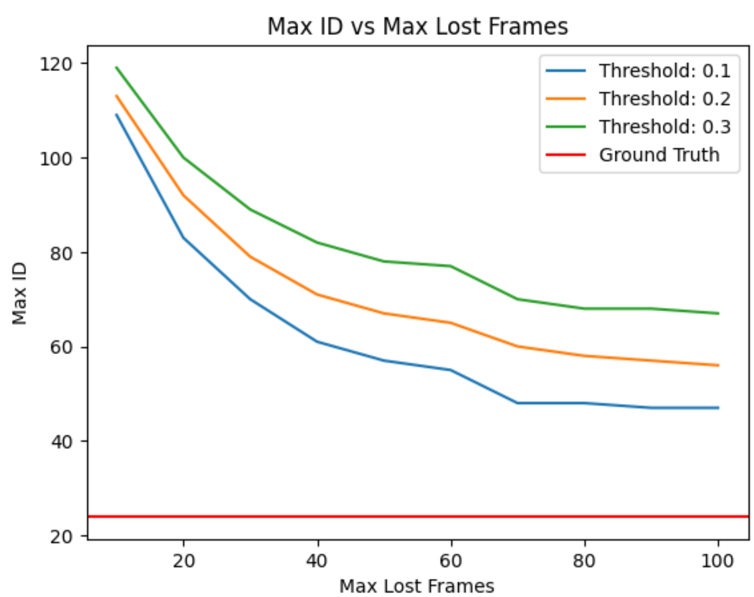

Figure 4: Parameter Search

I tried to find the best parameters to get as close as possible to the ground truth. However, I realized that it was impossible to achieve this using only IoU (as it does not account for objects passing each other). While having a high maximum frame value helps minimize ID switches, it also creates a problem where new objects are assigned old IDs instead of new ones. Additionally, this approach does not handle the issue of objects passing by each other. Setting a high threshold can reduce this problem but will lead to more IDs being created, as some objects may be detected as new due to detection inaccuracies.

3 Kalman-Guided IOU Tracking(Bounding-Box Tracker)

3.1 Objectives

The objective of this task is to develop add Kalman filtering on top of our iou based tracking.

3.2 Implementation

For each frame, the coordinates of each detected object are predicted before calculating the IoU. When objects are matched, instead of saving the new object as detected, I update the Kalman filter with the new coordinates.

3.3 Result

Figure 5: Kalman Filter Result

My detection is no longer working, and I can see that the Kalman filter predictions move independently in the background without even following an object. The IDs are no longer being maintained.

3.4 Challenges

I had some issues with matching the size of the matrices during the prediction and update steps of the Kalman filter (again). I manually calculated it to better understand the problem and managed to make it work.
Some of my predictions do not follow any object but instead track a path that the algorithm creates.

4 Appearence-Aware IoU-Kalman Object Tracker

4.1 Objectives

Implement object re-identification-based that will keep track of object even if they hide behind another.

4.2 Implementation

I kept the base from my previous assignment. For each frame, I compute features from the pixels within the bounding box. Then, using cosine similarity, I calculate a similarity matrix (in addition to the similarity matrix computed with IoU). I combine these matrices using coefficients, giving more weight to cosine similarity than IoU. After this, the implementation follows the same approach as in my second assignment.
The feature extraction is performed on patches taken from the bounding boxes. These patches are preprocessed to align with the input requirements of the deep learning model.

4.3 Result

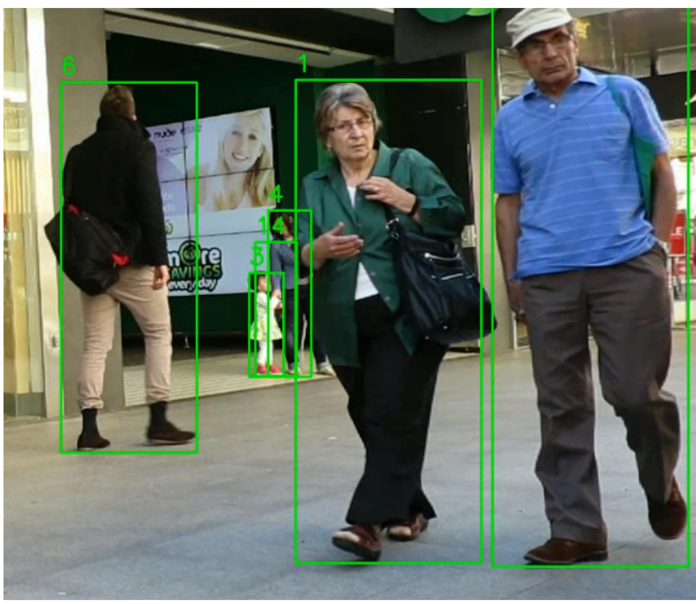

Figure 6: **Appearence-Aware frame 1**

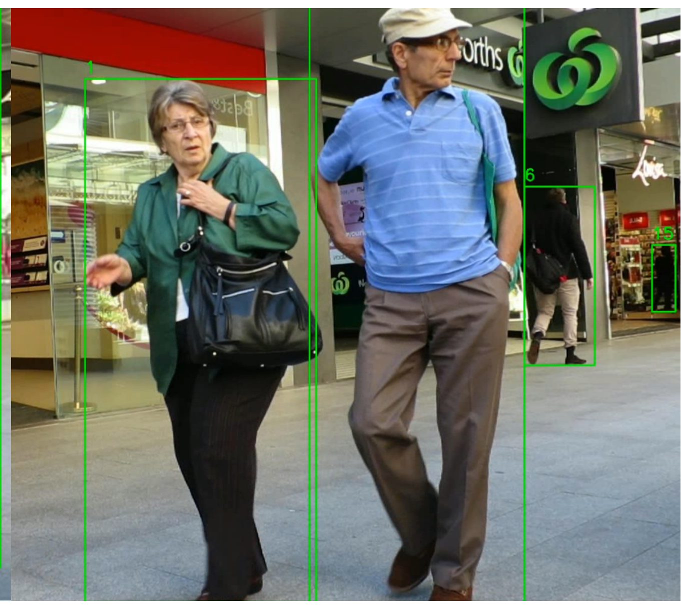

Figure 7: **Appearence Aware frame 2**

As we can see, ID 6 stays on the man in the background even if it passes behind the elderly couple.

4.4 Challenges

When one object moves in front of another, the algorithm sometimes exchanges their IDs. This issue arises because the similarity metrics (e.g., IoU or cosine similarity) cannot reliably distinguish between overlapping objects. Adjusting parameters, such as the coefficients of

the weighted matrix sum, can help address this problem.
Sometimes, objects are confused with each other when they are too close and share similar features, leading to an exchange of IDs. To resolve this issue, I tried changing the standard deviation and mean during normalization from constant values to variables calculated from the image crop of the bounding box.