ME5413 - Homework 1 Perception

By:
Anurag Roy - A0304443N

Task 1 - Tracking

Task 1.1 Using Template Matching

This task implements a basic object tracking pipeline using template matching and evaluates its accuracy using the Precision and Success Metric which will be explained below

Template matching is performed using OpenCV's cv2.matchTemplate() function with the cv2.TM_CCOEFF_NORMED method. This method measures the correlation between the template and the image while normalizing the values to account for brightness variations and produces a result between -1 and 1 where 1 indicates a perfect match and -1 indicates a negative correlation. T

The key steps are:

- Extracting the template (object region) from the first frame.
- Applying template matching to locate the object in each subsequent frame.
- Identifying the highest correlation location, which corresponds to the top-left coordinate of the detected bounding box.
- Storing the predicted bounding boxes as x, y,width and height for later analysis.

To visualize the tracking results:

- The predicted bounding box (GREEN) is drawn on the image.
- The ground truth bounding box (RED) is also drawn for comparison.
- The resulting images are saved for reference in **/data/seq1/results** for sequence 1 for example.
- The predicted bounding boxes are written to an output file in results/1_template_matching/trackresults_TM_seq1.txt .

Fig.1 Template Matching in Sequence 1

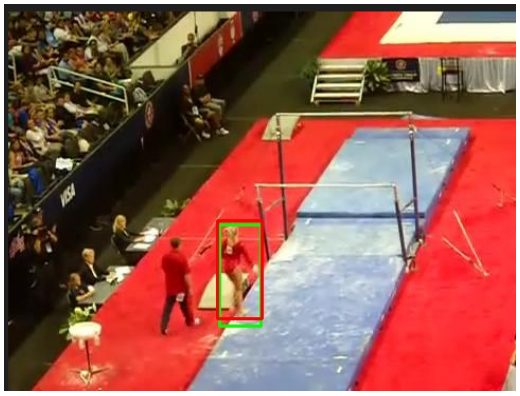


Fig.2 Template Matching in Sequence 2

Task 1.2 Object detection Algorithm and Association

This task consists of an approach using **DEtection TRansformer (DETR)** for object detection and **Intersection over Union (IoU)** as an association metric to track objects.

DEtection TRansformer (DETR) is a deep learning-based object detection model that eliminates the need for region proposal networks and directly predicts object bounding boxes and class labels using an attention-based mechanism.

Each frame in the sequence is passed through a pre-trained DETR model and then the model outputs bounding boxes, confidence scores, and class labels for detected objects. Then only the objects with a confidence score above a predefined threshold (e.g., 0.9 for sequence 1) are considered for tracking.

Since object detection happens independently in each frame, it is necessary to associate detected bounding boxes with objects from the previous frame. This is done using an **association metric**.

Intersection over Union (IoU) as the Association Metric

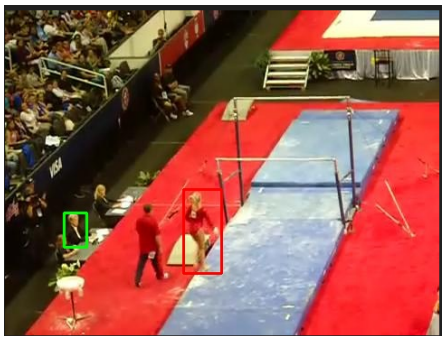
IoU is a widely used metric to measure the overlap between two bounding boxes:

$$\frac{Area\ of\ Intersection}{Area\ of\ Union}$$

The IOU ranges from 0 to 1 and for each detected object in the current frame, the bounding box with the highest IoU compared to the previous frame is selected.

Once the object detection and association are completed, the following steps are performed:

- The predicted bounding box (GREEN) is drawn on the image.
- The ground truth bounding box (RED) is also drawn for comparison.
- The resulting images are saved for reference in **/data/seq1/results2** for sequence 1 for example.The predicted bounding boxes are written to an output file in results/1_objectdetection_withassociation/trackresults_OD_seq1.txt



- Fig.3 Objection Detection in Sequence 1 Fig.4 Object Detection in Sequence 2

Task 1.3 Yolov11 Object Detection with an Extended Kalman Filter and IOU

This task details an improved tracking method that integrates the **YOLOv11 object detection model**, an **Extended Kalman Filter (EKF)** for motion prediction, and **Intersection over Union (IoU)** as an association metric. This hybrid approach enhances robustness against occlusions, noisy detections, and object appearance changes.

YOLO (You Only Look Once) is a state-of-the-art object detection framework designed for real-time performance. YOLOv11 has a transformer-based attention mechanism that improves the model's focus on critical regions within an image and an adaptive anchor-free object detection which is advantageous for detecting small objects.

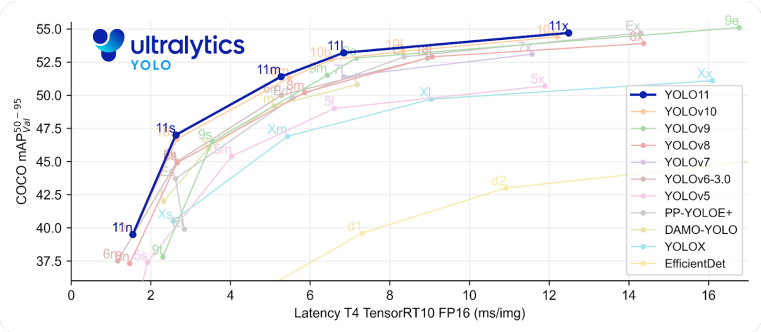


Fig.5 Comparison of objection detection models mAP based on the COCO Dataset.

Extended Kalman Filter

The sequences in the tasks introduce occlusions, missing detections, and noise. The Extended Kalman Filter helps by predicting the object's next position, ensuring smoother tracking.It is an enhancement of the Kalman Filter designed to handle nonlinear systems. While the standard Kalman Filter assumes linear dynamics, EKF extends its applicability to nonlinear motion models

The EKF tracks the object's **position and velocity** using a 6D state vector:

$$X = [x, y, w, h, v_x, v_y]^T$$

Where: x,y is object's position, w,h is the bounding box width and v_x v_y is the velocity components.

There is then a **state transition matrix** and a **noise variance matrix** to account for the movement and noises, and then there is a measurement update using the 6x6 Identity matrix to updates its prediction when a new bounding box is detected.

The process is as follows: The IoU is computed for the current and the previous frame object bounding box**.** If IoU > 0.3, the detection is considered a valid match; otherwise, tracking relies on EKF predictions.If no valid detections are found, the predicted state from EKF is used instead.

Fig.6 YOLO11+EKF+IOU in Sequence 1

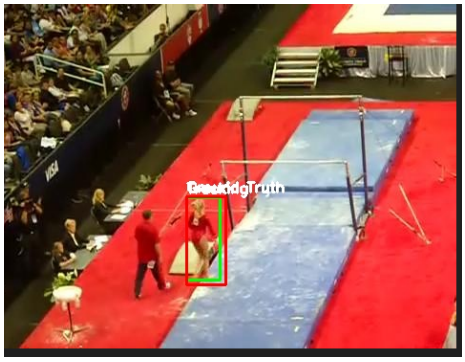


Fig.7 YOLO11+EKF+IOU in Sequence 2

Results

Precision (distance in pixels between the centers of the ground truth box and tracked bounding box) and **Success** (IoU of ground truth and tracked bounding box) are calculated. For **Precision**, the threshold chosen is 100 pixel as that gives the most wide values allowing for a better analysis.

Table1.2 Evaluation of SOT methods using Precision metrics

		Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Average Score
1	Template Matching	0.580	0.907	0.153	0.360	0.833	0.567
2	Detection with Association	0.460	0.673	0.273	0.113	0.880	0.480
3	Improved Method	**1.000**	**1.000**	**1.000**	**0.360**	**1.00**	**0.872**

Table1.1 Evaluation of SOT methods using Success metrics

		Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Average Score
1	Template Matching	0.331	0.477	0.115	0.221	0.603	0.349
2	Detection with Association	0.313	0.434	0.182	0.060	0.670	0.332
3	Improved Method	**0.877**	**0.718**	**0.669**	**0.091**	**0.771**	**0.625**

Discussion

The improved method of Yolov11 model + EKF + IOU achieves a higher average score in both Precision and Success metric compared to other two methods.

Generally, Yolo 11 is able to detect objects with higher accuracy than the DETR model, such as for example detecting the child even when he is bending downwards. Template matching also fails in this case as the template of the previous frame does not match the frames when the child bends over.

If an object disappears temporarily, EKF predicts its location quite well instead of losing track, for example in sequence when the player goes behind another player (occlusion) and all three YOLO, DETR and Template cannot detect the object anymore. Unlike IoU-only association, EKF predicts object movement even when detections are lost due to occlusions. Template matching definitely loses the objects since it is trying to match the frame of the first image to the new images where the object is in a different pose.

One drawback is that the improved method cannot handle **very** dynamic movements as shown through the Success and Precision score for sequence 4 as the volleyball player stays still for a few frames and then moves left very fast, and the detection falls onto another player for the rest of the sequence.

Task 2- Prediction

Task 2.1 - Constant Velocity Model(CVM)

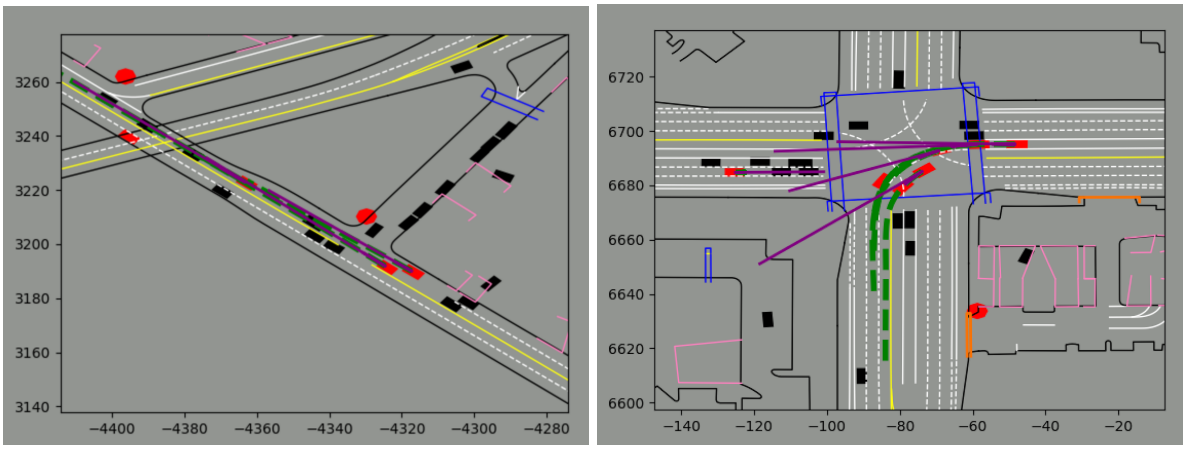
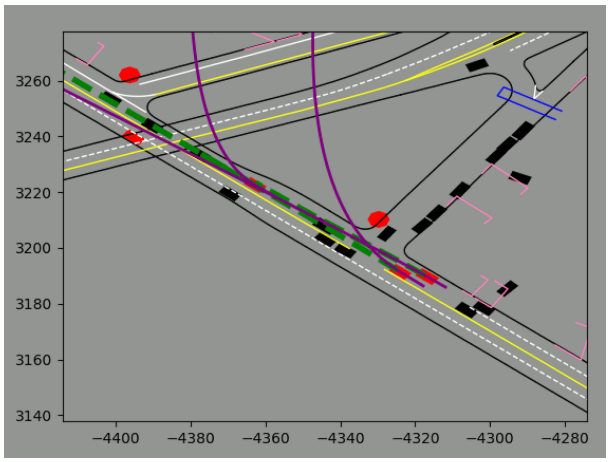


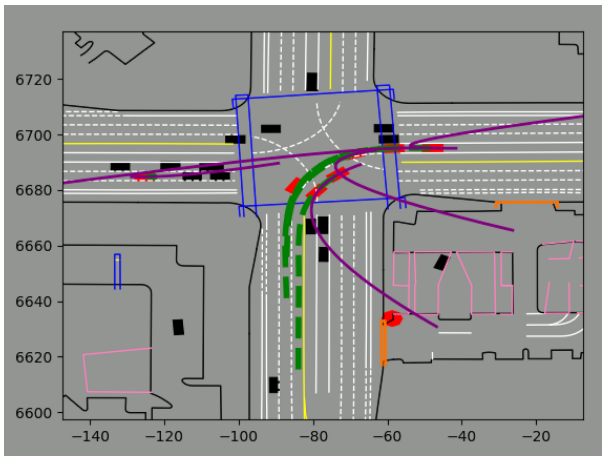
Fig 8. Trajectory Prediction using CVM - Predicted (Purple solid line) and Ground Truth (Green Dashed Line)

	3s	5s	8s
Average FDE	53.226	56.641	64.321
Average ADE	51.882	53.087	55.819

In the constant velocity model, we use the velocity at the 11th timestep to predict the future positions, and it works well for the linear motion as shown on the left image above. However it fails for turning agents and cannot predict curved paths, thus overshoots its trajectory. The small errors accumulate over the prediction horizon of 80 timesteps and make long term predictions unreliable.

Task 2.1 - Constant Acceleration Model(CAM)



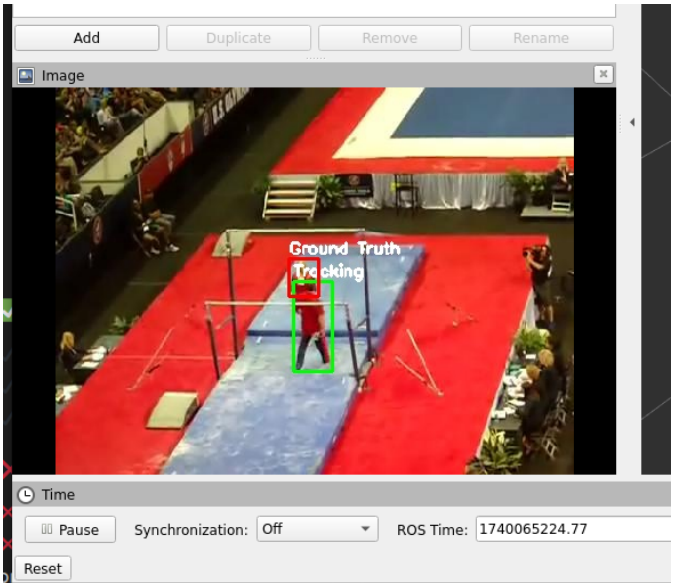


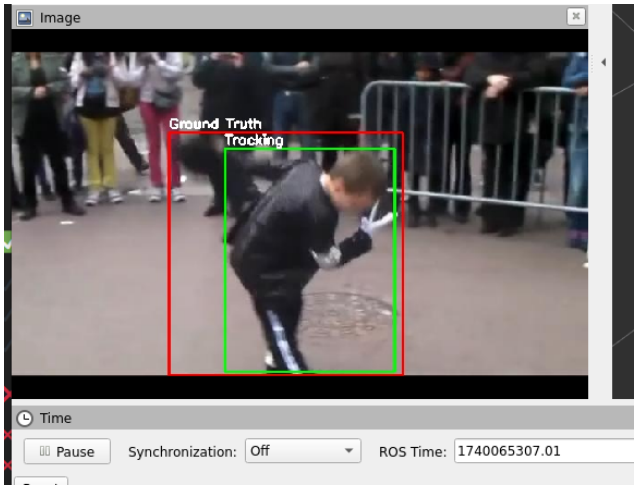
In the constant acceleration model, we use both the velocity and acceleration at time step 11 and assume constant acceleration over time, it handles acceleration and deceleration in dynamics and predicts the turning better too, but if the acceleration changes suddenly, predictions will deviate.

	3s	5s	8s
Average FDE	59.244	73.984	111.251
Average ADE	53.957	58.911	71.189

As shown above, the average FDE was quite much higher for CAM than for CVM, since the trajectory predicted by CAM would take drastic turns due to the acceleration prediction.

Task 3- Bonus Task





In /task3_bonus/ros_tracking_node.py there is the ros_tracking_node.py present with 4 ROS publishers that publish to topics, /me5413/viz_output, /me5413/groundtruth, /me5413/track, and /me5413/nusnetID with the appropriate msg types and information.

The visualization is shown above in rviz, and the rosbag file is present in /task3_bonus/results/4_rosbags for sequence 2 and sequence 3.

Task 4-Evaluation

Task 1 took the longest for me due to exploring a lot of different ways on how the improved method should work, this took me around 20 hours (4 days)

Task 2 took time in the beginning to understand the code but the implementation was simpler - 8 hours (2 days)

Task 3 - took 2 hours as I had prior experience with ROS

To run the code, please run the notebook normally with the environment created from **environment.yml**