ME5401 Autonomous Mobile Robotics

Homework1 Perception

Name: Yu Zhuoyuan

Email: yuzhuoyuan@u.nus.edu

AY2023/2024 (SEMESTER 2)

20th, Feb.,2024

Contents

Task 1 Single Object Tracking

Single Object Tracking (SOT) refers to the computer vision technique focused on monitoring the movement of a specific object across a series of frames in a video. This technology plays a critical role in various applications, including surveillance, autonomous vehicles, and human-computer interaction. It remains a dynamic field of research, with ongoing developments aimed at overcoming its inherent challenges. Integrating deep learning has opened new avenues for innovation, making SOT more reliable and versatile across various applications.

In this task, some basic standard single object tracking methods were adopted, and certain improvements were made based on these. However, traditional methods always face some problems that are difficult to resolve. The specific details are as follows:

1.1 Template Matching

Template matching is a technique used to find parts of a larger image that match a given template image. It is a fundamental method in the field of computer vision, widely applied in scenarios such as image recognition, surveillance, and video analysis.

The core idea behind template matching is to search for the area within the target image that is most similar to the template image. This process typically involves sliding the template image over the target image and, for every possible position of the template, calculating the similarity between the template and the area it covers. The area with the highest similarity is considered the matching area.

Based on this, I made my first attempt on *seq_1*. Referring to *firsttrack.txt*, I obtained the coordinates of the top-left pixel of the template, as well as the template's width and height, resulting in the template shown in Figure 1 below.

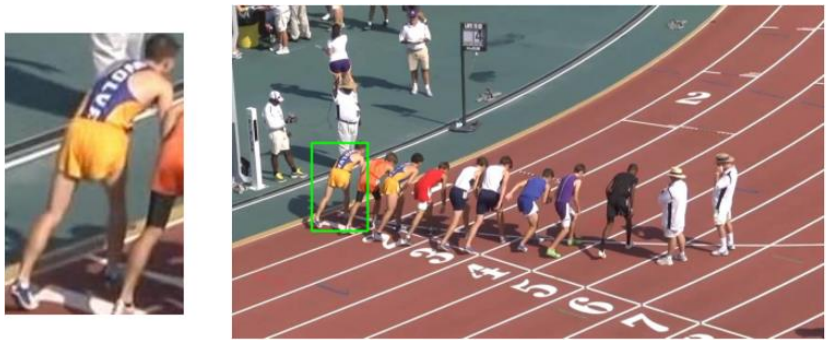

Figure 1 Template of seq_1

Afterwards, the cropped template is used to search within the entire scope of the image, identifying the location with the highest similarity to the template. However, this process has encountered some clear issues. Taking the first and second sequences as examples, the image selection meets the requirements when the athlete has not yet started running and the movement is minimal, or when a child walking does not cause significant rotation. But, when there is considerable movement or the subject turns to the side, incorrect selections often occur. This is especially true when there are many similar people or objects in the image, making it easy to mistakenly select a non-target location.

Attempts to improve:

In response to the issues mentioned above, I came up with two optimization ideas.

- Update the template in real time (Impractical)

The first one involves updating the template in real time to improve the similarity between the next frame and the template. Since the time interval between two frames is very short, even if there is a change in shape, it will not be significant. Hence, I used the bounding box recognized in one frame as the template for the next frame. However, in practice, this method proved to be unfeasible. When there is a noticeable movement or turn, using the previous frame as a template can still fail to timely capture the target's position, potentially leading to the selection of a non-target area. Moreover, since the template is updated in real time, the mistakenly selected non-target area would become the template for subsequent frames, rendering all following selections invalid. Therefore, this method is not viable.

- Limit the search range (Practical)

Another method involves restricting the search window to approximately 2 times the area around the bounding box selected in the previous frame. The advantage of this approach is that it can significantly reduce the chances of selecting non-target locations with high similarity, and greatly decrease the computational load, thereby increasing processing speed. Upon implementation, this method proved to be viable, substantially improving the precision of tracking. Hence, it was decided to use this method to optimize the basic template matching technique.

Accuracy

Ultimately, by employing the method of restricting the search window, we successfully optimized the basic template matching approach. The table below presents the results of the unoptimized algorithm versus the optimized algorithm, along with the Intersection of Union (IoU) data compared to the ground truth.

$$IoU = \frac{Intersection}{Union}$$

Table 1 The accuracy of basic and optimized template matching approaches

Method	Seq_1	Seq_2	Seq_3	Seq_4	Seq_5
Basic	0.501	0.284	0.337	0.398	0.285
Optimized	0.574	0.541	0.600	0.403	0.210

1.2 Kalman Filter

The primary function of the Kalman filter in image tracking is to predict and correct the position of objects across consecutive frames, enhancing the accuracy and stability of tracking. Its fundamental principle involves two main steps: prediction and update.

During the prediction phase, the Kalman filter forecasts the object's next state (including its position and velocity) based on its current state and the physical motion model. This prediction encompasses both the forecasted position and the uncertainty of the prediction (i.e., the predicted covariance).

During the update phase, when actual observations from the next frame are obtained (for example, the object's position detected by image processing algorithms), the Kalman filter uses this new information to update the object's state.

Based on this, we can derive the following formula:

$$\vec{x}_k = F\vec{x}_{k-1} + B\vec{u}_k + \vec{w}_k$$

In this, $\vec{x}_k$ represents the state matrix, F is the transition matrix, and $\vec{u}_k$ is the control input, indicating that the system allows for an external control quantity, which we will temporarily ignore. $\vec{w}_k$ is a random variable, known as process noise, representing the uncertainty of the state. In this task, we approximate the model as a constant velocity model, that is, Consequently, we can derive:

$$\vec{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x_k} \\ v_{y_k} \end{bmatrix} \qquad F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Afterwards, I utilized a Kalman filter to fit the image, using the values from template matching as observation data. From the visualization in the later section, it can be seen that the values from the Kalman filter are very close to those of template matching. Therefore, in practical applications, if the observer's data is relatively accurate, then the Kalman filter's prediction of future positions of objects based on the physical motion model is quite precise. Moreover, it can effectively extract useful signals from observation data with significant noise, enhancing the accuracy of position estimation.

1.3 Visualized results

In addition to Template Matching and the Kalman Filter, in exploring SOT, I also encountered many other mature algorithms, such as the MIL tracker and CSRT tracker, among others. In the final part, I listed the algorithms included in the OpenCV library and performed fitting. Figure 2 displays the IoU (Intersection over Union) ratio comparison of Template Matching, Kalman Filter, and another effective SOT algorithm with the ground truth for five sequences.

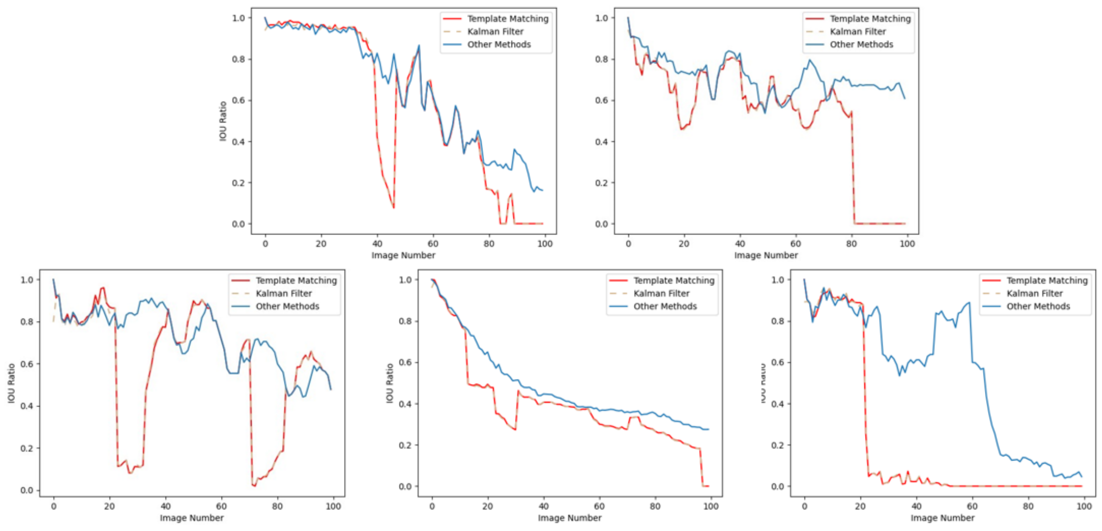

Figure 2 Results of different SOT algorithms

1.4 Discussion

The results show that these algorithms are generally more precise than the Template Matching algorithm in most situations. However, these algorithms are not capable of adapting to all scenarios; for example, some may lose track of the target when it undergoes shape changes due to rotation, while others struggle to track objects moving at high speeds. Therefore, I believe that to achieve better results with SOT, it is necessary to integrate it with deep learning to obtain optimal outcomes.

Task 2 Multi Object Prediction

Multi-object prediction (MOP) aims to accurately and in real-time track the motion trajectories of multiple targets from video sequences. This task holds significant value in various application scenarios such as video surveillance, autonomous driving, and robotic navigation.

Note: The Average Displacement Error (ADE) and Final Displacement Error (FDE) are shown in the tables while the detailed trajectories for every frame are listed in the coding part.

2.1 Constant Velocity Model

The Constant Velocity Model assumes that the velocity of the target remains constant over short intervals of time, allowing predictions of the target's future position based on its current location and speed.

$$x_{pred} = x + v_x \cdot \Delta t$$

$$y_{pred} = y + v_y \cdot \Delta t$$

Table 2 ADEs and FDEs(1s) of Constant Velocity Model

	1	2	3	4	5	6	7	8
ADE	1.70	1.33	1.47	1.38	1.37	0.95	0.80	0.62
FDE	0.89	1.93	2.52	2.01	1.64	1.42	1.17	1.04

Table 3 ADEs and FDEs(2s) of Constant Velocity Model

	1	2	3	4	5	6	7	8
ADE	1.58	2.26	2.83	2.29	1.82	1.50	1.30	1.10
FDE	1.92	4.42	5.79	4.23	2.94	2.62	2.38	2.04

Table 4 ADEs and FDEs(3s) of Constant Velocity Model

	1	2	3	4	5	6	7	8
ADE	2.15	3.83	4.70	3.47	2.60	2.11	1.88	1.61
FDE	2.87	9.36	10.77	7.17	5.44	3.95	3.61	3.07

2.2 Constant Acceleration Model

The Constant Acceleration Model assumes that the target's acceleration remains constant and unchanged. This model posits that although the position and velocity of the target may vary over time, its acceleration is constant meaning $v_{pred} = v + a\Delta t$.

Table 5 ADEs and FDEs(1s) of Constant Acceleration Model

	1	2	3	4	5	6	7	8
ADE	1.72	1.26	1.44	1.26	1.22	0.91	0.82	0.62
FDE	0.94	1.82	2.47	1.80	1.38	1.36	1.24	1.04

Table 6 ADEs and FDEs(2s) of Constant Acceleration Model

	1	2	3	4	5	6	7	8
ADE	1.63	2.18	2.77	2.07	1.69	1.46	1.39	1.12
FDE	1.92	4.42	5.79	4.23	2.94	2.62	2.38	2.04

Table 7 ADEs and FDEs(3s) of Constant Acceleration Model

	1	2	3	4	5	6	7	8
ADE	2.21	3.76	4.59	3.11	2.68	2.06	2.03	1.64
FDE	3.02	9.42	10.56	6.37	6.23	3.89	3.95	3.17

2.3 Constant Turn Rate and Velocity

The Constant Turn Rate and Velocity model incorporates the concept of turn rate to describe the motion of a target within a plane. The turn rate refers to the angle of directional change per unit of time, and a constant turn rate implies that the target's angular velocity remains unchanged.

Table 8 ADEs and FDEs(1s) of Constant Turn Rate and Velocity Model

	1	2	3	4	5	6	7	8
ADE	1.83	3.86	1.63	5.77	7.11	2.51	1.26	1.12
FDE	1.29	7.93	3.02	10.94	13.28	5.31	2.35	2.22

Table 9 ADEs and FDEs(2s) of Constant Turn Rate and Velocity Model

	1	2	3	4	5	6	7	8
ADE	1.96	7.82	3.83	11.68	13.86	5.75	2.90	2.64
FDE	2.74	14.75	9.08	23.13	26.47	11.30	6.41	5.89

Table 10 ADEs and FDEs(3s) of Constant Turn Rate and Velocity Model

	1	2	3	4	5	6	7	8
ADE	2.55	11.87	7.17	17.85	20.31	8.35	4.71	4.38
FDE	4.31	23.13	17.94	36.08	38.55	16.24	9.61	9.27

2.4 Discussion

According to the calculations and formulas, it is evident that the constant velocity model is suitable for scenarios with limited computing resources, featuring fewer model parameters and simple implementation, but it is not suitable for tracking acceleration or complex motion patterns. The constant acceleration model can more accurately predict the behaviour of targets in acceleration or deceleration movements, making it particularly applicable to scenarios with significant changes in motion patterns. The Constant Turn Rate and Velocity Model are better at understanding and predicting the dynamic behaviour of targets in space, especially suitable for tracking those targets that undergo both velocity and directional changes during movement, such as vehicles driving on roads.

Bonus Task Single Object Tracking in ROS

The task requires the use of the best-performing algorithm from Task 1, so I directly utilized the last algorithm to complete this task. The main objective here is to publish images from the sequence, ground truth data, and matriculation number through ROS. The images need to be converted using cv_bridge before subscription. Unlike Task 1, the function is defined as a callback function to avoid missing images published at the start.

Throughout this process, three nodes were constructed:

i. Execute Python code.
ii. Write data into a rosbag.
iii. Play the cropped images.