**Abstract**

**2.1 Research Question or Problem**

Due to recent progress in object detection, tracking-bydetection has become the leading paradigm in multiple object tracking（MOT）. Within this paradigm, object trajectories are usually found in a global optimization problem that processes entire video batches at once. The MOT problem can be viewed as a data association problem where the aim is to associate detections across frames in a video sequence.

Traditionally MOT has been solved using Multiple Hypothesis (MHT) or the Joint Probabilistic Data Association (JPDA) filters, which delay making difficult decisions while there is high uncertainty over the object assignments. These complexity of these algorithms increases exponentially with the number of tracked targets which makes them unsuitable for online tracking.

**2.2 Research Goals and Objectives**

Simple Online and Realtime Tracking (SORT) is a pragmatic approach to multiple object tracking with a focus on simple effective algorithms, but it returns a relatively high number of indentity switches. This is because the employed association metric is only accurate when state estimation uncertainty is low.

Our goal is to associate objects efficiently for online and realtime applications. And to get better accuracy and speed by using our algorithm to track through longer periods of occlusion,. Yet, the algorithm remains simple to implement and runs in real time.

**2.3 Research Design and Methods**

Detailed description of the code we download:

The code for deep\_sort is downloaded from the website:

<https://github.com/nwojke/deep_sort>.

**deep\_sort\_app.py**: main tracking code.

**generate\_detections.py**: Generate features for person re-identification, suitable to compare the visual appearance of pedestrian bounding boxes using cosine similarity.

**detection.py**: Detection base class. The attribute of the class is composed of the location bbox of the detection box, the confidence degree of the box and the feature of the box. Each detection box corresponds to each feature through the ZIP function.

detections contain every object detected.

**kalman\_filter.py**: A Kalman filter implementation and concrete parametrization for image space filtering.

**linear\_assignment.py**: This module contains code for min cost matching and the matching cascade.

**iou\_matching.py**: This module contains the IOU matching metric.

**nn\_matching.py**: A module for a nearest neighbor matching metric.

**track.py**: The track class contains single-target track data such as Kalman state, number of hits, misses, hit streak, associated feature vectors, etc. Each Tracker contains a tracks list.

**tracker.py**: This is the multi-target tracker class. It allows for matching, screening, updating operations based on detections that are detected:

Methodology the code used:

1. Track Handling and State Estimation:

Tracking scenario is defined on the eight dimensional state space (u,v,γ,h,x ̇,y ̇,γ ̇,h) that contains the bounding box center position (u,v), aspect ratio γ, height h, and their respective velocities in image coordinates. A standard Kalman filter with constant velocity motion and linear observation model, where we take the bounding coordinates (u,v,γ,h) as direct observations of the object state.

Tracks that exceed a predefined maximum age Amax are considered to have left the scene and are deleted from the track set. These new tracks are classified as tentative during their first three frames. Tracks that are not successfully associated to a measurement within their first three frames are deleted.

1. Assignment Problem

A match is a match between a currently active track and the current Detections. The matching degree combines motion metric and appearance metric.

motion metric between the i-th track and the j-th detection:



appearance metric between detection and track is the minimum cosine distance between feature vectors of Detection and detections contained within the track:

图示, 文本

描述已自动生成

1. Matching Cascade

When an object is occluded for a longer period of time, subsequent Kalman filter predictions increase the uncertainty associated with the object location. Therefore, authors introduce a matching cascade that gives priority to more frequently seen objects to encode our notion of probability spread in the association likelihood.

1. Deep Appearance Descriptor

Successful of deep\_sort requires a well-discriminating feature embedding to be trained offline, before the actual online tracking application. a pre-trained model can be used to generate features.

What we are going to utilize the code:

We will use the deep\_sort code as a base of our project to implement the multiple object tracking. Based on the MOT challenge benchmark evaluation, we will improve the accuracy by making some improvements as follows.

Improvement we plan to make:

1. Change a better algorithm for object detection.
2. Implement multiple classes of object detection and tracking.
3. Matching with not only the distance between objects but also their velocity and direction of motion.
4. Improve accuracy of detection and tracking when objects covering occurs.

Possible algorithms we will employ:

A better object detection algorithm rather than CNN such as YOLOv3 or YOLO-Fastest.

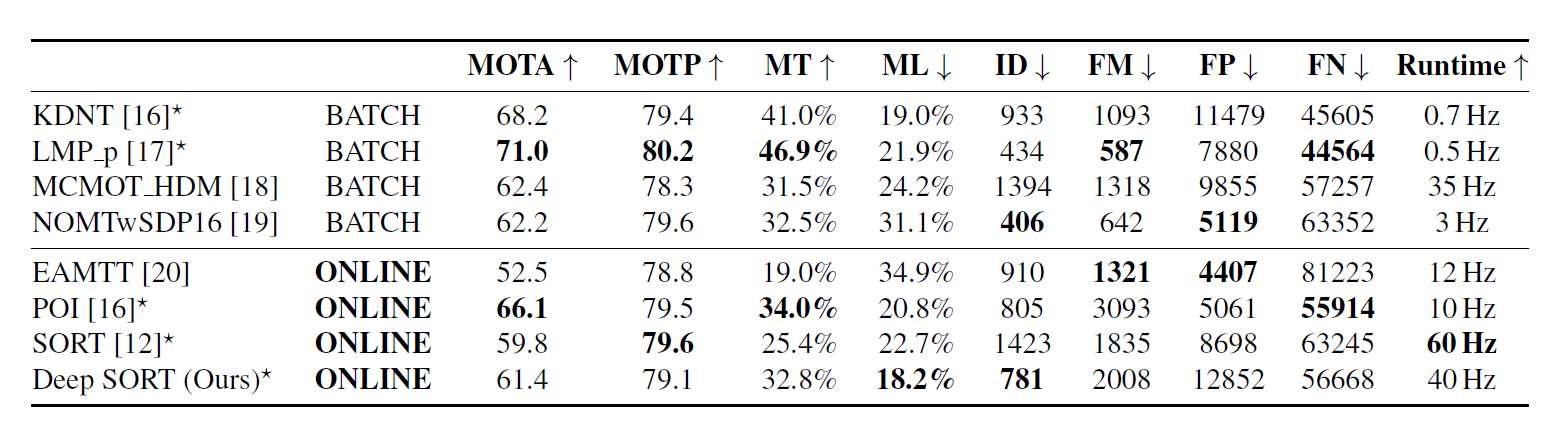
**2.4 Initial Results**

Since it is difficult to use one single score to evaluate multitarget tracking performance, we utilize the evaluation metrics defined in, along with the standard MOT metrics:

* Multi-object tracking accuracy (MOTA↑): Multi-object tracking accuracy.
* Multi-object tracking precision (MOTP↑): Multi-object tracking precision
* Mostly tracked (MT↑): Percentage of ground-truth tracks that have the same label for at least 80% of their life span.
* Mostly lost(ML↓) : number of mostly lost trajectories. i.e. target is not tracked for at least 20% of its life span.
* Identity switches (ID↓): Number of times the reported identity of a ground-truth track changes
* Fragmentation (FM↓): Number of times a track is interrupted by a missing detection.
* False positive(FP↓): number of false detections.
* False negative(FN↓): number of missed detections.

Evaluation measures with (↑), higher scores denote better performance; while for evaluation measures with (↓), lower scores denote better performance. True positives are considered to have at least 50% overlap with the corresponding ground truth bounding box. Evaluation codes were downloaded from[6].

Tracking performance is evaluated using the MOT16 benchmark. This benchmark evaluates tracking performance on seven challenging test sequences, including frontal-view scenes with moving camera as well as top-down surveillance setups. Compared with other algorithms, sort already achieve the higher MOTA score for the online trackers. And the deep sort algorithm goes further returns the fewest number of identity switches of all online methods while maintaining competitive MOTA scores, track fragmentations, and false negatives.



Most MOT solutions aim to push performance towards greater accuracy, often, at the cost of runtime performance. While slow runtime may be tolerated in offline processing tasks, for robotics and autonomous vehicles, realtime performance is essential. Generally achieve the best accuracy also tend to be the slowest. Deep SORT method combined the two desirable properties, speed and accuracy, with out the typical drawbacks. The Deep SORT method implementation runs at approximately 20Hz with roughly half of the time spent on feature generation. Therefore, given a modern GPU, the system remains computationally efficient and operates at real time.

Although deepsort has achieved good results, there is still space for improvement. On the one hand its excellent performance largely depends on the quality of detections. If the detection data is complicated, more complicated preprocessing may be required. On the other hand, only the distance relationship is used in the motion matching degree, not the real motion information. I think we can combine speed information to solve the Identity switches when similar people meet.

***2.5 Staffing Plan***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Background | Expertise | Role | Tasks |
| 11710211 | Yucheng Wang | Undergraduate | Department of Statistics | Leader and coder | The literature research; Code Running; Manage the timeline and check the work progress; Code for implementing multiple classes of object detection and tracking. |
| 11712531 | Tingting Zhang | Undergraduate | Department of Computer Science and Engineering | Coder | The literature research; Code Running; Code for implementing multiple classes of object detection and tracking. |
| 11710934 | Keyu Huang | Undergraduate | Department of Statistics | Coder | The literature research; Code Running; Code for changing a better algorithm for object detection. |
| 12032189 | Yuxi Liu | Graduate | Department of Computer Science and Engineering | Coder | The literature research; Code Running; Code for matching with not only the distance between objects but also their velocity and direction of motion. |

***2.6 Timeline***

The schedule for the final project on the course website:

Project Proposal Deadline: 10/21/2020(week 6)

First Survey Deadline: 11/25/2020(week 11)

Second Survey Deadline: 12/30/2020(week 16)

Project schedule and research plan:

Week 7: Try to run the downloaded code in our personal computer.

Week 9: Migrate our projects to the GPU.

Week 10: Change a better algorithm for object detection.

Week 11: The first Survey.

Week 13: Implement multiple classes of object detection and tracking.

Week 15: Matching with not only the distance between objects but also their velocity and direction of motion.

Week 16: The second Survey.

**Reference**

Write down the papers you refer to and the websites you download the codes.

The paper we referred to:

[1] N. Wojke, A. Bewley, and D. Paulus. “Simple online and realtime tracking with a deep association metric”, *arXiv preprint arXiv:1703.07402*, 2017.

The website we download the codes:

[2] <https://github.com/nwojke/deep_sort>

Other paper we referred to:

[3] N. Wojke and A. Bewley, “Deep cosine metric learning for person re-identification,” *in Proc. WACV. IEEE*, 2018, pp. 748–756.

[4] A. Bewley, G. Zongyuan, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in ICIP, 2016, pp. 3464–3468.

[6] L. Leal-Taix´e, A. Milan, I. Reid, S. Roth, and K. Schindler, “MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking,” arXiv preprint, 2015.

[7] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, “Poi: Multiple object tracking with high performance detection and appearance feature,” in ECCV. Springer, 2016, pp. 36–42.