Proposal for Final Project in Detection and Tracking of 2D/3D objects for self-driving Applications

A Proposal Prepared for the
Final Project of the graduate Course
CS405: Machine Learning
October 10, 2020

Prepared by:
Yucheng Wang 11710211*
Keyu Huang 11710934
Tingting Zhang 11712531
Yuxi Liu 12032189

Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, Guangdong, China 518055

* Team Leader

Abstract

This is our team's proposal for the CS405 final project—Detection and tracking of 2D/3D objects for self-driving applications. It includes seven components: (1) Research Question or Problem; (2) Research Goals and Objectives; (3) Research Design and Methods; (4) Initial Results; (5) Staffing Plan; (6) Timeline; and (7) Reference.

2.1 Research Question or Problem

Multiple Object Tracking(MOT) has always been a problem which is hard to proceed a satisfactory result balancing the accuracy and efficiency. Traditionally, MOT are solved by Multiple Hypothesis(MHT), the Joint Probabilistic Data Association(JPDA) filters and etc., which postpone in making difficult decisions when confronting high uncertainty over the object assignments. Also, the complexity of these algorithms increases exponentially as the number of the targets increase which makes them difficult to proceed online tracking.

Considering the facts mentioned above, distinct methods(DEEP_SORT, DEEP_SORT +YOLOv3) which could improve the performances will be implemented on our data set in this project. The results would be optimized by comparison.

2.2 Research Goals and Objectives

Firstly, an original data set(video clip) will be obtained from nearby sites of SUSTech for testing purpose.

The project will refer to previous papers about Deep_SORT and Deep_SORT+YOLOv3. Such two methods will be applied on our data set respectively to obtain distinct results. Comparisons between the results will be made and according to the characteristics for each model, optimization would be proceeded in order to mount the accuracy and efficiency for each model. Moreover, given that the main detection objects of the previous papers and researches are pedestrians, other detection objects(vehicles, traffic signs) will be also considered in our paper.

2.3 Research Design and Methods

We will first try Deep_SORT method and then combine Deep_SORT and YOLOv3 to gain a better result on our own data set. The codes and detailed descriptions of the codes we downloaded are as follow.

1. Codes and descriptions

(1.1) The code for deep_sort is downloaded from the website:

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

(1.2) Description:

1).deep_sort_app.py: main tracking code.

2).generate_detections.py: Generate features for person re-identification, suitable to compare the visual appearance of pedestrian bounding boxes using cosine similarity.

3).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.

4).kalman_filter.py: A Kalman filter implementation and concrete parametrization for image space filtering.

5).linear_assignment.py: This module contains code for min cost matching and the matching cascade.

6).iou_matching.py: This module contains the IOU matching metric.

7).nn_matching.py: A module for a nearest neighbor matching metric.

8).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.

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

(2.1) The code for Deep_Sort+YOLOv3 is downloaded from the website:

https://github.com/Qidian213/deep_sort_yolov3

(2.2) Description:

1).Detection: It is a detection class. The attributes of the class are composed of the position bbox of the detection frame, the confidence of the frame, and the feature of this frame. The zip function makes each detection box correspond to each feature. That is, detections contain all the objects which are detected.

2).Tracker: It is an object of the Tracker class. It is a tracker which can perform matching, filtering, and updating operations based on the detected detections.

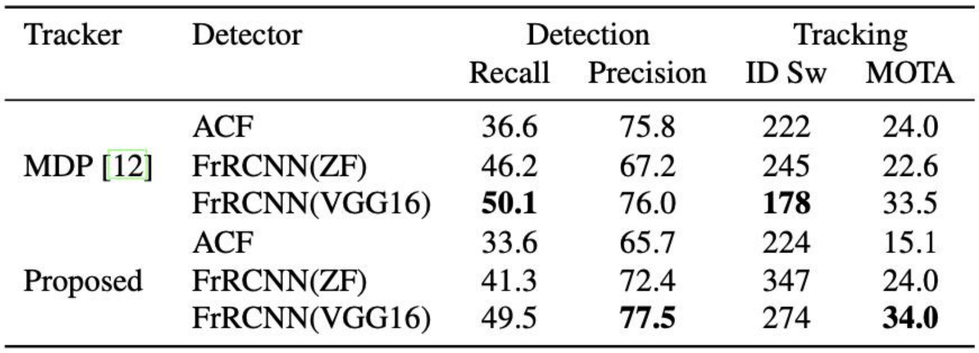
3).Track: It is an object of the Track class. Each Tracker contains a list of tracks. Each element of the list is an object of Track. Its attributes are the mean, variance, id, etc. generated by a single detection. It is a single detection in multiple tracking targets in the tracker.

3. Methodology of the code used:

(3.1) SORT

1). Detection:

SORT use FrRCNN detection method, FrRCNN is an end-to-end framework which consists of two stages based on convolutional neural network. The first stage extracts feature and proposes frames for the second stage which then classifies the object in the proposed frames. The advantage of this framework is that parameters are shared between the two stages creating an efficient framework for detection. Since the parameters are shared in both the two stages, such detection method is somehow efficient. Detection performance has a huge impact on the overall tracking effect. In the detection stage of the SORT, if we ignore other classes and only consider pedestrians, the detection results with output probabilities are greater than 50%.



Tracker	Detector	Detection		Tracking	
		Recall	Precision	ID Sw	MOTA
	ACF	36.6	75.8	222	24.0
MDP [12]	FrRCNN(ZF)	46.2	67.2	245	22.6
	FrRCNN(VGG16)	**50.1**	76.0	**178**	33.5
	ACF	33.6	65.7	224	15.1
Proposed	FrRCNN(ZF)	41.3	72.4	347	24.0
	FrRCNN(VGG16)	49.5	**77.5**	274	**34.0**

***proposed is SORT, FrRCNN(ZF), FrRCNN(VGG16) are modified FrRCNN.**

Figure 1.

2). Data Association

It uses the Hungarian assignment algorithm for data association. The cost matrix used is the IOU between the predicted position of the original target in the current frame and the target detection frame in the current frame. Of course, assignment results smaller than the specified IOU threshold are invalid. Using IOU can solve the problem of short-term occlusion of the target. This is because when the target is occluded, the occluder is detected, but the original target is not detected. It is assumed that the occluder is associated with the original target. Then after the occlusion is over, because the target IOU of a similar size is often larger, the correct association can be restored soon. This is based on the fact that the area of the occluder is larger than the target.

***IOU:Intersection over union**

3). Summary

The author uses Faster RCNN to detect the model, and uses Kalman filter to predict the state. The Hungarian algorithm based on the position of the detection frame and IOU makes the algorithm very efficient. However, such frequent ID switching is bad in practical applications.

(3.2) Deep_SORT

The previous SORT algorithm used a simple Kalman filter to process the correlation of frame-by-frame data and the Hungarian algorithm for correlation measurement. This simple algorithm achieved good performance at high frame rates. However, because SORT ignores the surface features of the detected object, it will only be accurate when the object state estimation uncertainty is low. In Deep SORT, we use more reliable metrics instead of correlation metrics, and use the CNN network in large scale of training on a large-scale pedestrian dataset and extracting features has increased the robustness of the network against loss and obstacles.

1). Track Handling and State Estimation:

Tracking scenario is defined on the eight dimensional state space $(u,v,\gamma,h,x^*,y^*,\gamma^*,h^*)$ 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 A_{max} 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.

2). 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:

$$d^{(1)}(i,j) = (\boldsymbol{d}_j - \boldsymbol{y}_i)^{\mathrm{T}} \boldsymbol{S}_i^{-1} (\boldsymbol{d}_j - \boldsymbol{y}_i),$$

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

$$d^{(2)}(i,j) = \min\{1 - \boldsymbol{r}_j^{\mathrm{T}} \boldsymbol{r}_k^{(i)} \mid \boldsymbol{r}_k^{(i)} \in \mathcal{R}_i\}.$$

3). 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.

4). 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.

(3.3) Deep_SORT+YOLOv3

The combination of Deep_SORT and YOLOv3 algorithm to achieve target tracking. It is equivalent to the structure of two stages in target detection. It adopts detection + track and does not perform end-to-end framework which is in SORT. The advantage is that we can separately perform detection part (YOLO) and track part (Deep_SORT) according to the tracking effect in the actual project.

4. Utilization of the code:

We will use the deep_sort and deep_sort+YOLOv3 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). modify a better algorithm for object detection not only for pedestrian, but for more objects which might occur in self-driving context.

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.

5). Try to obtain a model which could lowest the delay, however with satisfactory accuracy.

5. Possible algorithms we will employ:

A better object detection algorithm rather than CNN and YOLOv3, maybe YOLO-Fastest.

MOTDT algorithm.

.

2.4 Initial Results

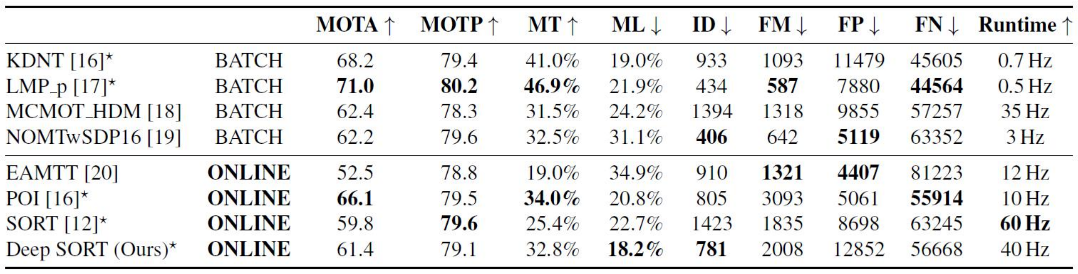
Since it is difficult to use one single score to evaluate multitarget tracking performance, we utilize the evaluation metrics defined in, along with the standard MOT metrics:

- 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.



		MOTA ↑	**MOTP ↑**	**MT ↑**	**ML ↓**	**ID ↓**	**FM ↓**	**FP ↓**	**FN ↓**	**Runtime ↑**
KDNT [16]*	BATCH	68.2	79.4	41.0%	19.0%	933	1093	11479	45605	0.7 Hz
LMP_p [17]*	BATCH	**71.0**	**80.2**	**46.9%**	21.9%	434	**587**	7880	**44564**	0.5 Hz
MCMOT_HDM [18]	BATCH	62.4	78.3	31.5%	24.2%	1394	1318	9855	57257	35 Hz
NOMTwSDP16 [19]	BATCH	62.2	79.6	32.5%	31.1%	**406**	642	**5119**	63352	3 Hz
EAMTT [20]	**ONLINE**	52.5	78.8	19.0%	34.9%	910	**1321**	**4407**	81223	12 Hz
POI [16]*	**ONLINE**	**66.1**	79.5	**34.0%**	20.8%	805	3093	5061	**55914**	10 Hz
SORT [12]*	**ONLINE**	59.8	**79.6**	25.4%	22.7%	1423	1835	8698	63245	**60 Hz**
Deep SORT (Ours)*	**ONLINE**	61.4	79.1	32.8%	**18.2%**	**781**	2008	12852	56668	40 Hz

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, dataset obtaining.
11710934	Keyu Huang	Undergraduate	Department of Statistics	Coder	The literature research; Code Running; Code for changing a better algorithm for object detection, dataset obtaining.
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, dataset obtaining.

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

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

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

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

[4] 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.

[5] 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.

[6] https://github.com/nwojke/deep_sort

[7] https://zhuanlan.zhihu.com/p/114349651