2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)

Effective Vehicle Tracking Algorithm for Smart Traffic Networks

Shuai Hua
Computer Engineering
San José State University
San José, CA, USA
shuai.hua@sjsu.edu

David C. Anastasiu*
Computer Engineering
San José State University
San José, CA, USA
david.anastasiu@sjsu.edu

***Abstract*—A smart traffic analysis system could be used to reduce congestion, prevent accidents, as well as to control traffic flow. Such a system would need to make use of many technologies, such as computer networking, communication, image processing, object detection, and tracking. In this paper, we introduce an efficient vehicle tracking algorithm which could be used to help create a smart city traffic control system. Our algorithm is targeted to solve the multi-object tracking (MOT) problem. Our method follows a track-by-detection paradigm, i.e., it relies on vehicle detection results to decide whether a detected vehicle in sequential frames belong to the same track. Our vehicle tracker extends the intersection over union (IOU) tracker and improves upon it by fusing historical tracking information, taking into consideration the balance between tracking efficiency and effectiveness. We demonstrate the effectiveness and efficiency of our approach using the UA-DETRAC benchmark dataset. Our proposed method runs at an average speed of 1,264 frames per second. We conclude that our tracker could be useful for applications running in a real-time environment.**

***Index Terms*—smart traffic, vehicle tracking, intersection over union, localization, smart city.**

I. Introduction

The maturity of the Internet and hardware breakthroughs have recently made it possible to use machine-learning in industry applications. One useful application of machine learning would be to develop a traffic control system that is able to think and act like a human being. For this application, the first priority is to train a machine to recognize as many objects as possible, track their location changes over time, and then predict changes in aggregate traffic patterns. Recently, many researchers have successfully developed methods to recognize (i.e., detect) diverse objects, e.g., cats, books, computers, people, and vehicles. At the same time, much research has been launched to track moving objects, including vehicles and pedestrians. There are many practical scenarios to which vehicle tracking can be applied. Smart traffic surveillance systems, for instance, highly depend on vehicle detection and tracking algorithms to help a police officer analyze traffic conditions so as to control traffic flow. A fast running and highly accurate tracker can also be used to infer movement statistics for vehicles from video data, such as the current speed of cars on the road; the officer would be able to supervise traffic conditions by means of this information.

Generally, a tracking algorithm is given as input a segment of video data; it analyzes the video, and then generates the tracking results by associating a unique ID to the same vehicle in each of the video frames the vehicle is found in. The sequence of vehicle positions in frames for a uniquely identified vehicle is also known as a *track*. A candidate track, which may only contain a subset of the vehicle positions or possibly a subsequence of the track, is called a *tracklet*. There is exactly one vehicle in a track or a tracklet. Ideally, we expect the ID of the vehicle to stay unchanged, from the first frame in which it emerges in the video to the end of the last frame, when it disappears from the video.

The tracker should be able to assign different IDs to different vehicles in the video. Obviously, vehicle tracking is a complicated problem. We normally solve this problem by comparing and matching the unique features of each vehicle across the video frames. We expect that the same vehicle will still have the same unique features across multiple frames, which can be used to differentiate it from other vehicles. We also make the assumption that the vehicle moves at a constant speed, and the location will not change abruptly between consecutive frames. However, in reality, there are still many challenges to overcome in tracking algorithms, since keeping features unchanged across frames is nearly impossible. Common features that an algorithm may use to detect recurring vehicles are the color of the object and the shape and size of the object. Unfortunately, because of lighting conditions (illuminations), camera rotation, or vehicles changing direction of travel, most of the time these features are not constant across frames. Moreover, the point of view of the camera plays a big role in how a vehicle is perceived. For example, one can easily imagine that the size of the vehicle gets smaller and the color becomes unstable as the vehicle moves away from the camera, towards the horizon.

Another critical problem that the tracking algorithm should deal with is the occlusion caused by other vehicles and non-vehicle objects. This issue is especially prominent when traffic is heavy: one vehicle is hidden in some frames by another vehicle that is driving closely in front of it and then appears again later. We expect the tracker to have the capability to identify these types of scenarios and correctly track a vehicle, even if it is lost in traffic for some time. Even when using a static camera recording traffic in the same position and rotation over time, the quality of the tracking result could vary greatly under different weather conditions. In general, a tracker will likely perform better on sunny days than in rainy conditions. Our aim is to develop a tracker that works well in all normal

* Corresponding Author

978-1-7281-1442-2/19/$31.00 ©2019 IEEE
DOI 10.1109/SOSE.2019.00019

weather conditions.

In the following sections, we will first review two tracking methodologies, known as track-by-estimation paradigm and track-by-detection paradigm, we will introduce the UA-DETRAC benchmark, then we will present our tracking algorithm, and finally we will summarize the experiment result.

Our research contributions are as follows: we trained the well-known YOLO [21] object detector as a general vehicle detector and measured its effectiveness on the UA-DETRAC dataset [22]; we evaluated the tracking efficiency and effectiveness of two recent tracking algorithms on the dataset – the IOU [7] and the MDP [23] trackers; we developed a novel tracker that extends the IOU tracker by taking into consideration historical IOU data; finally, we show that our tracker achieves a good balance of tracking effectiveness and efficiency performance.

II. Related Works

In this section, we will review some tracking approaches following the prediction-correction and track-by-detection paradigms.

A. The Prediction-Correction Algorithm

The prediction-correction formulation is usually based on statistical estimation and assumption. One of the most typical approaches is based on the theory of Kalman filters, which has seen wide use in many real-world applications [6], [8], [9], [10], [18], [19]. However, most of the authors derive this algorithm by using complex mathematical representations that are sometimes too complicated for the novice to understand. In this section, we are going to review papers by Pei et al. [19] and Faragher [10]. Pei et al. gave a straightforward definition of Kalman filtering: it is an algorithm that combines two imprecise estimations, one which comes from the prediction, and the other which is generated by the measurement. The algorithm fuses these two estimations linearly to obtain a more precise result. One advantage of this article is that it derives a complex mathematical theory from simple concepts, helping readers that do not have a strong mathematical background better understand Kalman filtering.

In the second article [10], Faragher presented an example that demonstrates the simple and intuitive idea behind deriving the Kalman filter. Faragher's publication is especially targeted for the reader who does not have a strong mathematical background. Faragher used car movement as his practical example, which is pertinent to our research domain. Readers that have a basic understanding of Newtonian movement should be able to quickly grasp the Kalman filter approach via this simple example.

B. The Track-by-Detection Algorithm

Advances in object detection technology provide another approach to solve the object tracking problem, through a method known as track-by-detection. This method has been mainly used in the tracking of pedestrians and vehicles. Much research has been published on the problem of object detection. Naphade et al. released a vehicle detection benchmark, which is known as NVIDIA AI City Challenge [16], [17], in order to encourage more researchers to get involved in building a smart traffic system. General object detectors, such as the you only look once (YOLO) detector [20], have been successfully used by teams competing in the 2017 AI City Challenge to accurately localize and classify traffic-related objects, such as cars, buses, trucks, motorcycles, bicycles, pedestrians, and traffic lights [5]. Farhadi and Redmon [21] further improved the efficiency of YOLO by using multi-scale predictions without specifying the anchor boxes. Fergus and Zeiler [24] presented a way to visualize and understand convolutional neural networks (CNNs), which are the core of most deep learning-based detection algorithms. Ahmed et al. proposed a CNN to solve the image re-identification problem [1].

Object tracking research has also gained popularity recently. Hua et al. proposed an approach for vehicle speed estimation in the 2018 NVIDIA AI City Challenge [11] that also follows the track-by-detection paradigm. They relied on YOLO as the vehicle detector of choice, and used the Lucas-Kanade tracker [15] for establishing vehicle tracks. Bochinski et al. published an intuitive idea for the vehicle tracking problem, known as the IOU tracker [7]. The authors mainly used the bounding box information provided by the detector to track the vehicle. This benefits efficiency and makes it possible to be used in a variety of pragmatic applications. In this paper, we used this algorithm as one of our baselines.

Bewley et al. implemented a simple online and real-time tracking algorithm based on the IOU tracker, called SORT [4]. The tracking algorithm follows a similar track-by-detection framework and identifies the tracking problem as a data association problem. In the SORT algorithm, tracked features include the central point of the bounding box, the ratio between the bounding box width and height, as well as the object movement speed, which is usually assumed to be constant.

Xiang et al. formulated the tracking problem as a decision making problem [23]. The authors relied on the Markov decision process to solve the problem. Hence, they named their method the MDP tracker. Our paper will also use this algorithm as one of our baselines.

Kalal et al. proposed a tracking algorithm called median flow (MF) [12]. The authors used the forward-backward error (FB error) algorithm to detect the failure in the tracking task. In their paper, the authors first explained the general idea and gave the mathematical derivation for the FB error. Then, they demonstrated the idea by tracking a single point and then extended it to tracking multiple points. Particularly, the authors built their model based on the MF tracker, which was itself originally invented based on the Lucas-Kanade tracker [14]. On the basis of the Lucas-Kanade tracker, the MF tracker requires the FB error of the points between two consecutive frames to be less than a threshold, which is normally 50%, as the median flow name indicates. Points with an error rate greater than this threshold will be removed.

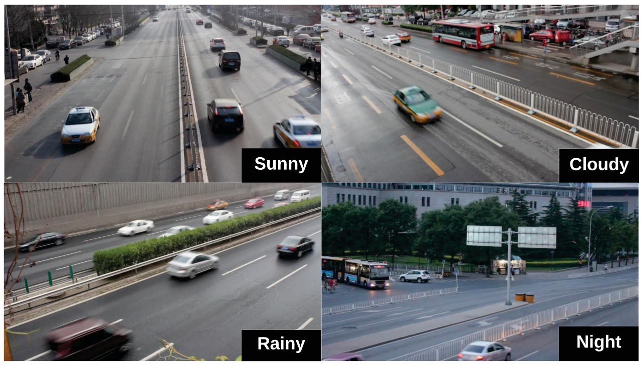


Fig. 1. An example of the UA-DETRAC dataset under different conditions.

Kalal et al. also developed a novel *tracking-learning-detection* framework on the detection and tracking task, known as TLD [13]. As indicated by the paper title, "Forward-Backward Error: Automatic Detection of Tracking Failures," the authors pointed out that there have to be three independent components to successfully build a TLD tracking algorithm: the tracker, the learner, and the detector. One of the creative contributions in this paper is the idea of positive-negative (P-N) learning for the detection. Kalal et al. used the random forest classifier to simulate the P-N learner. The tracker in TLD is based on the MF results, which they improved on by adding a failure detection algorithm to reliably tackle the fast occlusion problem which commonly happens in object tracking. Moreover, they used the FB error to identify the failure of a trajectory.

In addition to the tracking algorithms, many researchers also worked on the evaluation of tracking performance. Bashir and Porikli [2] presented a metric to evaluate object detection and tracking systems in early 2006. Bernardin and Stiefelhagen [3] also developed the evaluation for the Multi-object tracking (MOT) system, which is known as the CLEAR metric. Wen et al. described the state-of-the-art MOT system protocol [22], which they called the UA-DETRAC MOT benchmark. We evaluated the performance of the baseline trackers and our method by using this protocol.

III. The UA-DETRAC Benchmark

In this section, we focus on reviewing the UA-DETRAC benchmark which we used for vehicle tracking and evaluation. Wen et al. released the UA-DETRAC annotated traffic dataset, which they described in their "UA-DETRACK: A new benchmark and protocol for multi-object detection and tracking" article [22], and they also introduced a new protocol to evaluate the MOT system, which they called the UA-DETRAC MOT protocol.

A. Dataset

This dataset includes a total of 10 hours of video segments recorded at 24 different locations in China, with a resolution of 960×540 and a speed of 25 fps. The total number of annotated objects is 1.21 million, consisting of 140,000 frames and 8,250 vehicles. The annotated objects include car, bus, van, and other, where "other" represents some low-resolution regions within the image. According to the weather conditions, illuminations, occlusions, and traffic conditions, the dataset is also partitioned into three different levels: easy (10 sequences), medium (20 sequences), and hard (10 sequences). We illustrate a snapshot of the UA-DETRAC benchmark in Fig. 1.

TABLE I
2D Ground Truth Label File

Frame	Top	Left	Width	Height
0	0.96	0.29	0.049	0.087
0	0.87	0.24	0.044	0.078
1	0.77	0.29	0.057	0.091
1	0.27	0.30	0.069	0.091

For each annotated image, there exists one corresponding text file containing ground truth annotations. An example of such a ground truth file is given in the Table I. Each line of this file is used to indicate one object and contains 5 columns separated by a space. The first column represents the vehicle type, including car, van, bus, and other. The vehicle type is represented by a number starting from 0. The last four columns are used to store the 2D coordinate of the bounding box of that object, including the top-left corner coordinates, the width, and the height of a bounding box, relative to the size of the image in pixels.

B. Evaluation Protocol

Wen et al. proposed an approach that jointly evaluates the detector and the tracker and generates scores indicating the overall performance of the MOT system [22]. This has been widely known as the UA-DETRAC benchmark.

1) Detector Evaluation: The UA-DETRAC protocol uses the precision versus recall (PR) curve to learn the performance of the detector. Another metric that could be used to measure detection performance is average precision (AP). The higher the AP score is, the better the detector's performance is as well.

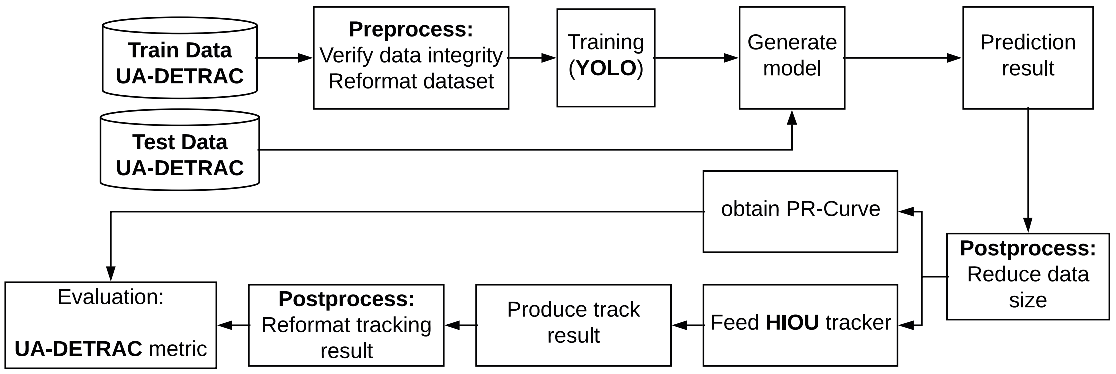
2) Tracker Evaluation: There are different indicators describing the performance of the tracker from variable viewpoints. We will briefly explain each metric in this section.

Mostly track (MT) is a number used to count the ratio between the ground truth tracks and the predicted tracklets with a length of at least $m\%$ of the length of the predicted track. In our work, we set $m = 80$.

Similar to the MT, mostly lost (ML) represents the total number of trajectories in which the percentage of the track that is correctly predicted by some tracklet is less than $l\%$, where $l = 20$ in our work.

Identity switches (IDS) is a number used to count the number of ID changes. A perfect tracker should have IDS=0.

The Fragmentation (FM) score is defined as the percent of tracks that were fragmented.



Train Data UA-DETRAC
Preprocess: Verify data integrity Reformat dataset
Training (YOLO)
Generate model
Prediction result
Test Data UA-DETRAC
obtain PR-Curve
Postprocess: Reduce data size
Evaluation: UA-DETRAC metric
Postprocess: Reformat tracking result
Produce track result
Feed HIOU tracker

Fig. 2. Pipeline of IOU tracker by YOLO detection.

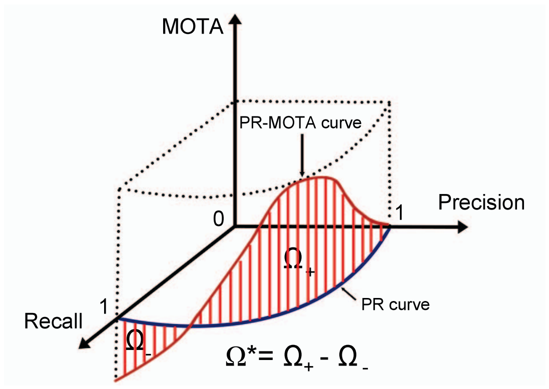


Fig. 3. PR-MOTA curve. Ω^* is a value describing the overall performance of the MOT system.

Multi-object tracking accuracy (MOTA) of a certain sequence is defined as

$$MOTA = 100 \times \left(1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t}\right),$$

where t represents a certain frame, and FN_t and FP_t are the false negative and false positive rates for that frame, respectively. GT_t represents the number of ground truth instances in that frame. Particularly, by analyzing the fraction term, one can see that it is a ratio that accounts for all potential errors during the tracking. Notably, if IDS is greater than 0, the ratio may be greater than 1 and, therefore, the MOTA value may be less than 0 in some cases. If evaluating the tracker performance on multiple sequences, MOTA is defined as

$$MOTA = 100\times\left(1 - \frac{\sum_v \sum_t (FN_{v,t} + FP_{v,t} + IDS_{v,t})}{\sum_v \sum_t GT_{v,t}}\right),$$

where v indicates that the score will be accumulated across all videos.

Multi-object tracking precision (MOTP) is defined as

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t},$$

where t represents a certain frame, c is the total number of matching bounding boxes, and d denotes the difference between the predicted object i with the ground truth object. This metric does not provide information about the tracker; it only indicates the performance of the detector.

3) Comprehensive Evaluation: As mentioned in the previous section, the goal of the evaluation is to find a way to consider the detection and tracker results jointly. Using one of the above metrics alone is not enough. The UA-DETRAC protocol joins the PR curve detector metrics with the tracking metrics to generate a comprehensive evaluation result. Typical UA-DETRAC metrics include the PR-MOTA score, the PR-MOTP score, the PR-IDS score, the PR-MT score, the PR-ML score, and the PR-FM score. We are particularly interested in the PR-MOTA curve, which can be used to evaluate the overall performance among trackers. A typical PR-MOTA curve is shown in Fig. 3. In this figure, the red curve is the PR-MOTA curve, while the blue curve is the precision-recall curve used to rank the performance of the detector.

IV. Vehicle Tracking using the IOU Tracker

In this section, we will provide more details about the IOU tracker, since our idea is motivated by this algorithm. As its name (intersection over union) indicates, Bochinski et al. [7] mainly used the IOU between bounding boxes in two consecutive frames to associate the objects in their algorithm. They assumed that the detector is able to perform well in each frame. In this situation, it is reasonable to assume that two bounding boxes belonging to the same object in two consecutive frames will have a high IOU score. Given a certain IOU threshold σ during the tracking, the tracker computes the IOU between two bounding boxes — one is in the new frame and the other is from the previous frame — and identifies the target by looking at the best match IOU above the threshold. The IOU score is computed as

$$IOU(a, b) = \frac{Area(a) \cap Area(b)}{Area(a) \cup Area(b)}. \quad (1)$$

Bochinski et al. also proposed other parameters to further improve the performance of the tracker. These parameters include the maximum confidence score α, the minimum confidence score β, and the shortest tracklet length γ. Because the authors did not consider any image information in their algorithm, the method is able to outperform others, resulting in very high tracking speeds.

A. Maximum Confidence Score

One of the hyper-parameters in this approach is the maximum confidence score α. With this hyperparameter, at least one detection in a tracklet should have a confidence score greater than this threshold. The authors used this parameter to guarantee that at least one detection in the track is the true positive detection. We applied different maximum confidence scores in our experiment, ranging from 0.0 to 0.9, in increments of 0.1.

B. Minimum Confidence Score

Similar to the maximum confidence score described above, the authors used the minimum confidence score β to filter out false alarms. All detections in the tracklet should have a confidence score which is greater than this threshold. In our experiment, we ranged the minimum score from 0.0 to 0.9, in increments of 0.1.

C. Shortest Tracklet Length

The authors used the length of the tracklet to indicate the number of consecutive frames in which a particular vehicle has been successfully tracked. In any tracking problem, the minimum tracklet length γ should be 1 frame. In our work, we fixed this value to be 2.

D. IOU Between Two Consecutive Frames

As the authors expounded in the paper "High-Speed Tracking-by-Detection Without Using Image Information" [7], the value of the IOU score plays a key role in this tracking algorithm. It can be used to find out the same instance of a vehicle between two consecutive frames by assuming spatial invariance. We ranged the IOU threshold from 0.3 to 0.8 in our experiments, in increments of 0.1.

V. Methodology

The pipeline of our track-by-detection HIOU tracker is shown in Fig. 2.

A. Vehicle Detection by YOLO

In our method, have to train a detector to predict the vehicles in order to implement a tracker following the track-by-detection paradigm. We used the YOLO algorithm [21] as our vehicle detector, which has been shown to be a very good detector for traffic-related objects [5].

B. Pre-processing the Detection Result

When setting the confidence threshold to 0.0, the size of the prediction output is 42 GB. We found that this predicted result includes an excessive number of false alarms, which in turn have very small confidence scores and significantly slow down tracking. In order to increase tracking efficiency, we chose a minimum confidence score of 0.0001 to approximate the ideal threshold 0.0, which reduced the total size of the detection result to only 1.3 GB.

C. Historical Based IOU Tracker

Motivated by the IOU tracker mentioned above, we developed a history-based IOU (HIOU) tracker, which has the capability to overcome the detection of false alarms. Bochinski et al. [7] used the overlap between two consecutive frames to associate a new detection with an object in the previous frame if their overlap is greater than a threshold. This algorithm works well if the detector is able to generate high accuracy predictions; however, their approach becomes ineffective when there is interference. Occlusion is one of the typical noises in vehicle tracking. Normally, occlusion happens when the tracked vehicle is hidden by another vehicle, or hidden by a non-vehicle object. Unfortunately, our tracker is unable to handle the occlusion caused by a vehicle, because it lacks image information. However, our tracker's advantage over the IOU tracker is in its ability to deal with the interference of non-vehicle occlusions. We will demonstrate this in the next section.

Algorithm 1 HIOU Tracker

```
1: **Input**
2:     D = $\{D_0^I, D_1^J, ..., D_{F-1}^K\}$ = $\{\{d_0^0, d_0^1, ..., d_0^{I-1}\}, ...,$
3:         $\{d_{F-1}^0, d_{F-1}^1, ..., d_{F-1}^{K-1}\}\}$, where $d_i^j$ = $(b_i^j, s_i^j, id_i^j)$.
4:     $\alpha \leftarrow$ max confidence score
5:     $\beta \leftarrow$ min confidence score
6:     $\gamma \leftarrow$ min track length
7:     $\eta \leftarrow$ max backward frame, $\theta \leftarrow$ min IOU, ID $\leftarrow$ 1
8:     for $d_0^j \in D_0^I$ do
9:         $id_0^j$ = ID and ID $\leftarrow$ ID + 1 only when $s_0^j > \beta$
10: End
11: Start
12:     for f = 1 to F-1 do
13:         for $d_f^j \in D_f^J$ do
14:             for $d_{f-1}^i \in D_{f-1}^I$ and $s_f^j \geq \beta$ do
15:                 if IOU($b_{f-1}^i$, $b_f^j$) $\geq \theta$ then
16:                     $id_f^j \leftarrow id_{f-1}^i$
17:                 else
18:                     $d_f^j \rightarrow$ untrack
19:         if length(untrack) $> 0$ then
20:             for $d_f^m \in$ untrack and max(0, f-$\eta$-1) $\leq \eta' \leq$ f-2 do
21:                 try to match $d_f^m$ with historical frame $\eta'$
22:         if length(untrack) $> 0$ then
23:             for $d_f^m \in$ untrack do
24:                 assign new ID: $id_f^m \leftarrow$ ID, ID $\leftarrow$ ID + 1
25:     for f = 0 to F-1 do
26:         aggregate $d_f^i$ by the ID and save to $track_i$
27:     for $track_i$, where i $\in$ [1, max(ID)] do
28:         if max score($track_i$) $< \alpha$ then
29:             remove $track_i$
30:         if length($track_i$) $< \gamma$ then
31:             remove $track_i$
32: End
```

We mainly use the IOU to associate the bounding boxes and do not introduce any image information in our proposal; therefore, we can simplify the complicated vehicle tracking task to a bounding box association problem. We differentiate our algorithm with the IOU tracker by considering tracking history in solving the association problem. Similar with the

IOU tracker, we only track those detections in which the confidence scores are above a certain threshold. We use this threshold to prevent adding false alarms to the tracker and increase tracking accuracy. The IOU tracker assigned a new ID to the detection *immediately* when it finds that the overlap is less than the threshold. This might lead to some mistakes in some circumstances, such as occlusions, or detection deviations, meaning a detector may only predict part of the object. These two interference scenarios will cause the overlap between two consecutive frames to drop slightly below the threshold; consequently, the IOU tracker will separate the track into two or more tracks. Moreover, it is common to have the detector generate false predictions. In this case, the IOU tracker is unable to realize the detection failure and will split a track into multiple tracks. Fortunately, our algorithm is able to fix these issues. When there is a car that failed to be detected in a previous frame, or that has an overlap score *slightly* below the threshold, our method will continue to compare the target frame with at most η historical frames to see if it is possible to link the current detection with some detections in the earlier frames. To ensure robustness, our method slightly decreases the IOU threshold proportional with the history distance (number of intermediate frames) between the current frame and the historical frame being searched, as follows:

$$\theta^{'} = \theta - 0.1, \; where \; \min_{\theta} \geq 0.3. \tag{2}$$

Our method assumes that the value of the IOU score varies linearly, and that very small overlap (below 0.3 in our experiments) may not indicate a good match. Our method is described in Algorithm 1. We use D_i^J to indicate one of the detections, where i represents the frame ID, beginning from 0, and J indicates the total number of detections within that frame. Similarly, the jth object in frame i is denoted as d_i^j, where d_i^j is made up of the bounding box, the confidence score, and the vehicle id, which are aggregated as (b_i^j, s_i^j, id_i^j). Next, we will provide more details about our algorithm. Note that we initialize the hyper-parameters of our tracker in lines $2-5$. When processing the first frame, our method also assigns unique IDs to the detections in that frame. It is then reasonable to start tracking from the second frame. The method first iterates across each detection to compute the overlap with all detections in the previous frame and tries to associate the best matched bounding box among the ones being compared. If it is unable to find a matched detection among previous detections, it will store this detection in a cache called *untrack* (lines $10-13$). After finishing this stage, if there exist any bounding boxes in *untrack*, the method starts the trace back process by looking at the historical frames (lines $14-16$). Finally, if it is still unable to match the bounding box with one in earlier frames, it assigns a new ID to the detection.

We further improve the performance of our tracker with two other parameters. We can tune the length of the tracklet γ to get rid of some false alarms. For example, it is meaningless to have a tracklet with a length of less than 2 frames in reality. In order to increase the quality of the track, we can also use the maximum confidence score α and minimum confidence score β to filter out false alarms.

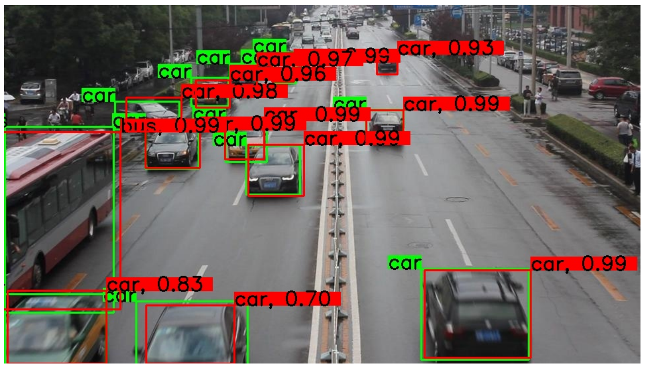


Fig. 4. An example of the visualization of the YOLO detection results versus ground truth annotation.

VI. Experiments

We executed the model training and vehicle prediction on a server equipped with an Intel i7 2.8 GHz CPU, 16 GB memory, and an NVIDIA Titan Xp GPU. We used another server equipped with 2 Intel(R) Xeon(R) E5-2680 v3 CPUs and 384 GB memory to run the tracking experiments for our tracker.

In our experiment, we used YOLO to localize vehicles in the UA-DETRAC benchmark dataset. We split the annotated training set with a ratio of 8:2 samples. In order to equally evaluate the performance of the detector, we intentionally kept a similar distribution of weather conditions in our training and test sets. Our training dataset includes 48 sequences with 67,745 images. It took our method 50.35 hours to finish the training. We used the remaining 12 sequences to evaluate the performance of the detector. The method spent 11.12 hours to generate vehicle detections. We also reserved 4 sequences with 4,451 images to evaluate the performance of the tracker. Fig. 4 depicts an example of the detection and ground truth bounding boxes. In this figure, we plotted the detected vehicles with their type and confidence score on the top-right side of the bounding box in red and the ground truth vehicle type on the top-left side of the bounding box in green.

We generated the precision-recall curve by evaluating the training sequences, test sequences, and track sequences. According to the UA-DETRAC MOT metric, we set the IOU between the prediction and ground truth bounding box to 0.7. We generated the precision recall curve by changing the threshold of the confidence score, ranging from 0.0 to 1.0, in increments of 0.1. Each curve was plotted using 11 different precision-recall pairs. Fig. 5 shows the precision-recall curves among the training sequences (left), test sequences (middle), and track sequences (right).

We chose the UA-DETRAC MOT evaluation protocol to evaluate the tracking performance of our tracker. In order to learn how the hyper-parameters affect the performance of the

TABLE II
COMPREHENSIVE PERFORMANCE OF HIOU, IOU, AND MDP TRACKER

Tracker	PR-MOTA	PR-MOTP	PR-MT	PR-IDS	PR-FM	PR-MOTAL	PR-ML
			Overall (Sunny, Night, Cloudy, Rainy)				
HIOU	**7.83**	**8.53**	**8.90**	**1.39**	**4.23**	**8.12**	**0.21**
IOU	6.48	8.23	6.81	12.24	16.14	6.59	0.40
MDP	6.55	7.97	7.55	3.79	10.28	6.56	0.42
			Sunny				
HIOU	8.08	8.85	7.01	1.03	6.73	8.10	0.71
IOU	7.07	**9.23**	6.94	2.37	3.80	7.22	0.73
MDP	**8.12**	8.86	**8.34**	**0.63**	**2.33**	**8.13**	**0.52**
			Night				
HIOU	**7.72**	**11.11**	6.58	3.89	7.22	**7.82**	0.57
IOU	6.68	10.60	6.67	4.91	6.73	7.06	0.62
MDP	7.35	9.74	**8.69**	**1.72**	**4.23**	7.38	**0.50**
			Cloudy				
HIOU	**6.21**	**6.49**	6.81	**0.51**	**3.07**	**6.23**	**0.17**
IOU	5.31	6.34	**6.82**	3.91	4.73	5.47	0.19
MDP	4.65	6.27	6.65	0.93	**3.07**	4.65	0.34
			Rainy				
HIOU	**10.21**	10.61	**10.95**	**0.34**	4.40	**10.24**	**0.27**
IOU	9.24	**10.64**	6.46	1.58	1.50	9.37	0.45
MDP	10.07	10.26	8.30	0.37	**1.18**	10.08	0.40

In the table, note that, for the PR-MOTA, PR-MOTP, PR-MT, and PR-MOTAL metrics, higher values are better. For the PR-IDS, PR-FM, and PR-ML metrics, lower values represent higher tracker performance. The best performing result for each metric has been highlighted in bold.

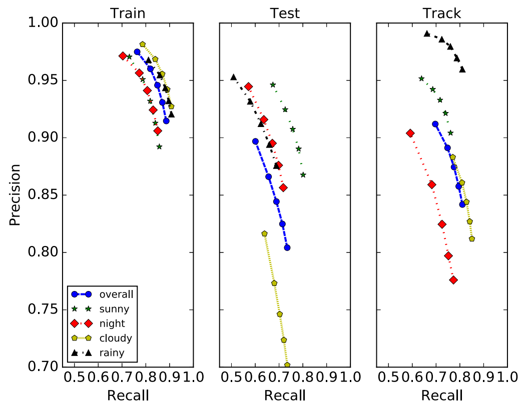


Fig. 5. Precision-recall curve under different weather conditions.

tracker, we ran the experiment multiple times. For the HIOU tracker and the IOU tracker, we ranged the minimum confidence score between 0.0 and 0.9, the maximum confidence score between 0.5 and 0.9, and fixed the minimum track length to 2 frames. For the MDP tracker, we ranged the confidence score between 0.0 and 0.9. For both trackers, we ranged the IOU score between 0.3 and 0.8, in increments of 0.1. For the historical length parameter in our HIOU tracker, we tested with a length of 3 frames.

We show a typical tracking result for the HIOU, IOU, and MDP trackers in Fig. 6. Furthermore, Fig. 7 illustrates the capability of trackers to overcome the non-vehicle occlusion. In this figure, there is a non-vehicle occlusion, which is a pole. At the top of the figure, we can see that the pole had little impact on our HIOU tracker (note the vehicle with ID 124), while it caused the vehicle ID to change from 29 to 38 in the middle figure, in which the vehicle was tracked using the IOU tracker. The bottom figures demonstrate that the MDP tracker is able to overcome this type of occlusion as well (note the vehicle with an ID 55).

In order to obtain comprehensive evaluation scores, we first have to compute the precision-recall values by changing the minimum confidence score threshold. Then, with these values, we are able to generate comprehensive metrics by using the UA-DETRAC MOT evaluation toolkit. In our experiments, we obtained multiple results by applying different hyper-parameters, and selected the best results for each method, which we include in Table II.

According to the definition of the PR-MOTA, PR-MOTP, PR-MT, and PR-MOTAL metrics, the higher these values, the better the tracker. For metrics PR-IDS, PR-FM, and PR-ML, lower values represent higher tracker performance. Table II shows that the overall performance of our proposed tracker is better than that of the IOU and MDP trackers. In other words, the performance of the IOU tracker has been improved using historical tracking information. We also highlight the best performing method according to each metric using bold script. Our tracker outperforms the IOU tracker in most metrics, even though in some metrics the MDP performs the best. Considering method efficiency results described in Table III also, we conclude that the HIOU tracker can balance tracking efficiency and effectiveness, resulting in the best oveall performance.

We also used the metrics MT, MOTA, MOTP, IDS, FM, FAR, and ML without considering the performance of the de-

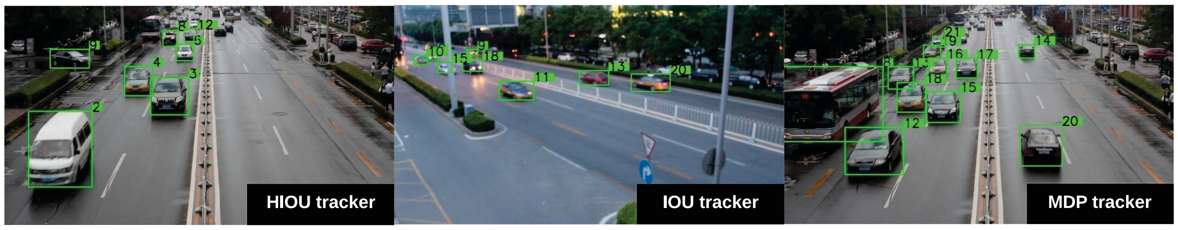


Fig. 6. Visualization of the tracking results among three trackers.

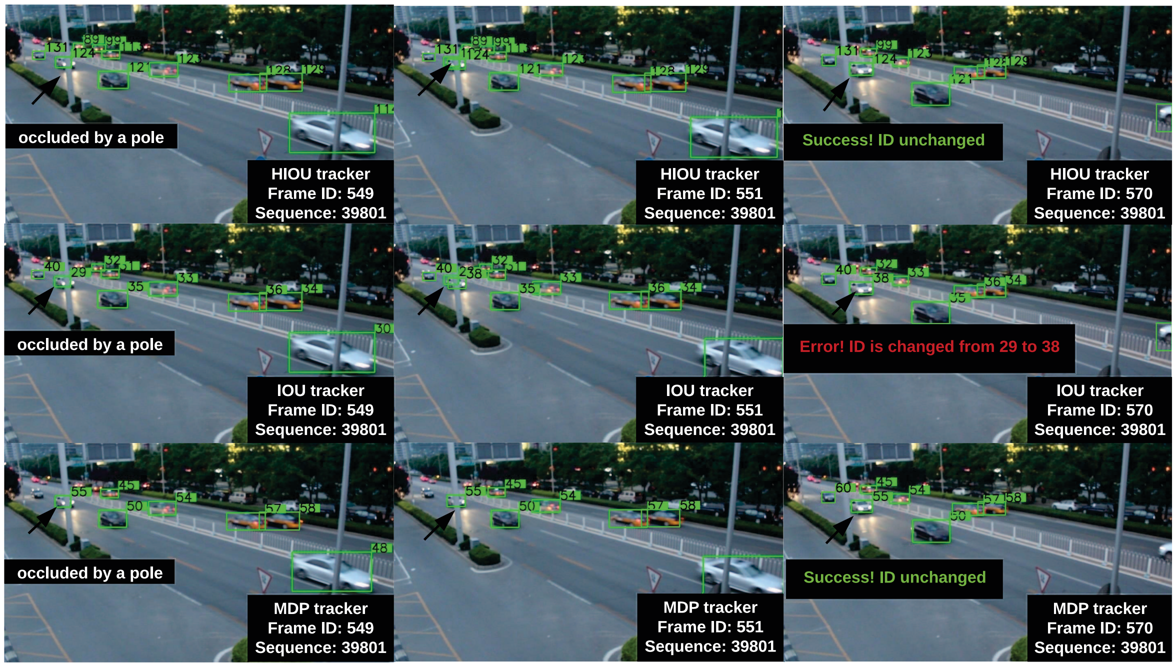


Fig. 7. The capability of trackers to overcome non-vehicle occlusion (a pole).

tector. We visualized the relationship between the confidence score and the metric MT, MOTA and MOTP in Fig. 8. In this figure, we fix the maximum confidence scores of the HIOU tracker and the IOU tracker to 0.6. We varied the minimum confidence scores for the HIOU tracker and the IOU tracker, and varied the IOU threshold for both trackers from 0.5 to 0.8. We found that the best tracking performance can be obtained with the IOU threshold set to 0.5 and the confidence score set to 0.7. Additionally, we were interested in how the IOU threshold affects these tracking metrics. We analyzed this relationship in Fig. 9. We note that the IOU threshold affects result quality for both trackers, since the overlap plays an important role in the HIOU tracker and IOU tracker. One should note that setting the IOU threshold greater than 0.6 will drop the most track (MT) performance.

Runtime efficiency is one of the most critical metrics to evaluate a tracker. We have to balance tracking accuracy and speed when we apply the algorithm in a practical environment. The speed of the tracker is measured in processed frames per second (fps). We compared the speed of the HIOU tracker with that of the IOU tracker and show these results in Table III. Results show that the IOU tracker outperforms our tracker with regards to effectiveness by a small factor, relative to the efficiency difference between our tracker and the MDP tracker.

TABLE III
RUNTIME EFFICIENCY (FPS), GIVEN CONFIDENCE SCORE OF 0.5

Trackers	20012	39801	40131	63525
HIOU	1,162.15	1,558.61	890.23	3,285.86
IOU	**1,973.97**	**2,607.94**	**1,870.67**	**3,623.36**
MDP	2.74	3.79	2.58	7.37

VII. FUTURE WORK

Our work mostly considered the trade-off between tracking efficiency and effectiveness. We ultimately implemented an approach to solve the vehicle tracking task without adding any image information. We improved the existing IOU algorithm by adding historical tracking data into the process, resulting in a tracker that matches the effectiveness of the MDP state-

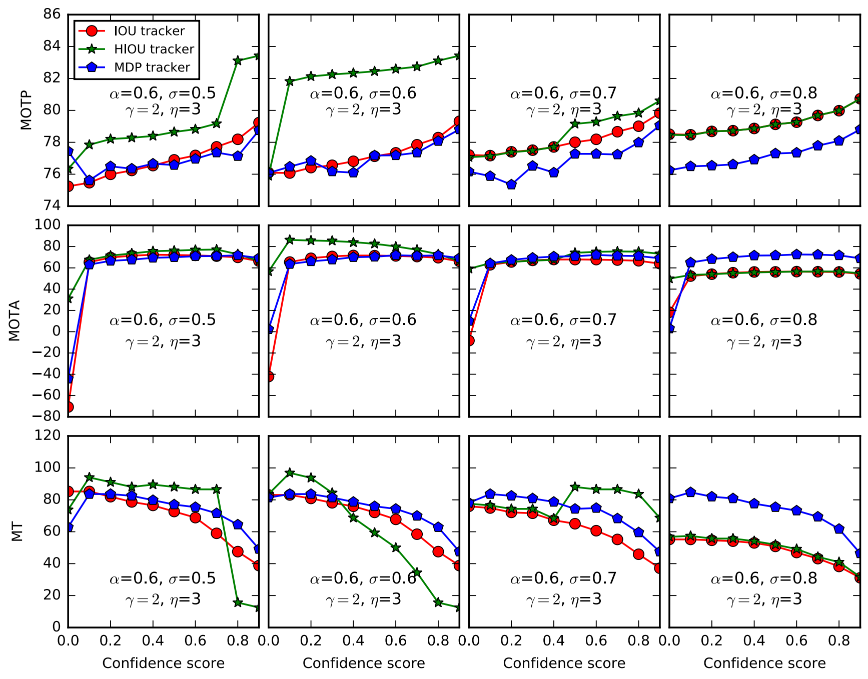


Fig. 8. The overall performance in 4 sequences among the HIOU, IOU, and MDP trackers.

of-the-art tracker at a speed similar to that of the naïve IOU tracker. Our model is able to handle occlusions caused by detection false alarms; however, our model is unable to tackle occlusions caused by another vehicle because it does not use image information. One possible future improvement would be to create a new tracker that utilizes some image information, such as the shape, color, and appearance of the vehicle. It would be interesting to see if this could be used to decrease the number of ID switches. One will need to carefully choose image features that will not be detrimental to the speed of the tracker.

We only relied on one well-known detector in our work. Therefore, we were unable to evaluate the impact of that detector on the tracker. However, many research experiments suggest that the quality of the detection will impact the performance of the track-by-detection trackers. Consequently, it is possible to generate a higher quality tracker by enhancing existing vehicle detection approaches. We plan to re-execute our experiments with multiple detectors and quantify the effect of detection quality improvement on tracking effectiveness.

VIII. Conclusion

Our target was to develop a simple, yet fast, tracker with relatively high tracking performance, and also make it easy to understand and usable in real MOT tasks. For this, we implemented a history-based IOU tracker (HIOU), which is an extension and optimization of the IOU tracker. We followed the track-by-detection methodology to develop our tracking algorithm. Our HIOU tracker is able to overcome minor detection false alarms by looking further back in history than the IOU tracker. Even without using image information, our method achieved high tracking performance and relatively high speed compared to two other baseline trackers.

We relied on the state-of-the-art UA-DETRAC dataset and its evaluation protocol to measure the effectiveness of our compared methods. In order to localize vehicles, our methods used the existing and well-known YOLO object detector. Our method achieved a relatively high PR-MOTA metric score compared to the IOU tracker. It is significant to note that the overall ID switch score in our tracker is much lower than that of both baseline trackers.

We formulated a complicated vehicle tracking problem as a simple data association task, without considering any image information. We hope this work will inspire others to implement a faster, more accurate tracker that can be better used in solving real problems. In addition, while convolutional neural networks have revolutionized object detection technology in recent years, we hope this work will further motivate researchers to improve the performance of object detectors, which will further improve our ability to track vehicles. These methods will then become the cornerstone of the intelligent traffic control systems of tomorrow.

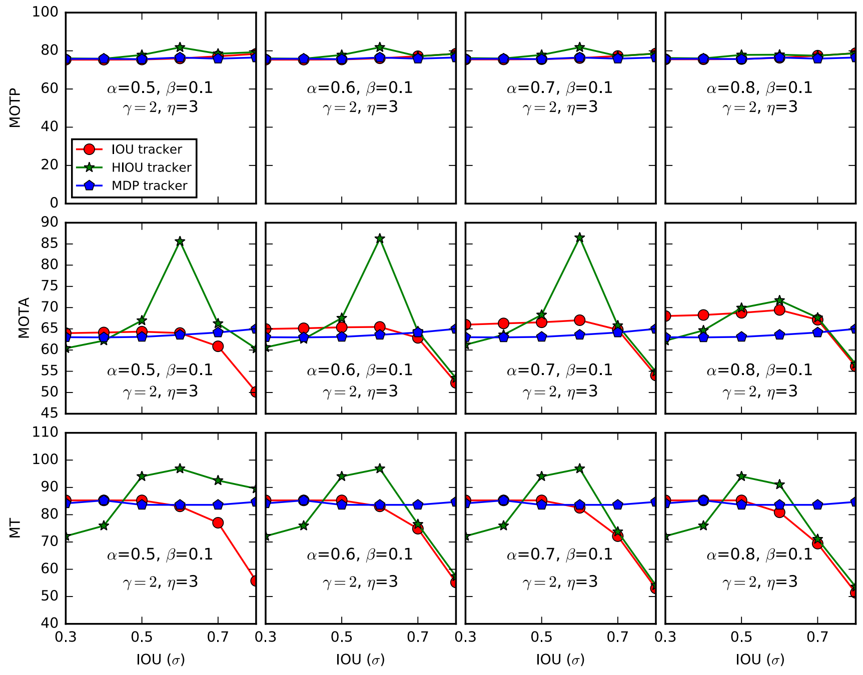


Fig. 9. The overall performance in 4 sequences among the HIOU, IOU, and MDP trackers.

References

[1] E. Ahmed, M. Jones, and T. K. Marks. An improved deep learning architecture for person re-identification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3908–3916, June 2015.

[2] F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. In *Proceedings 9th IEEE International Workshop on PETS*, pages 7–14, 2006.

[3] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, May 2008.

[4] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[5] N. Bhandary, C. MacKay, A. Richards, J. Tong, and D. C. Anastasiu. Robust classification of city roadway objects for traffic related applications. In *2017 IEEE Smart World NVIDIA AI City Challenge*, SmartWorld'17, Piscataway, NJ, USA, 2017. IEEE.

[6] G. Bishop, G. Welch, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-3175):59, 2001.

[7] E. Bochinski, V. Eiselein, and T. Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2017.

[8] F. Dellaert and C. Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on Intelligent Transportation Systems*, volume 1, 1997.

[9] Y. Du and F. Yuan. Real-time vehicle tracking by kalman filtering and gabor decomposition. In *2009 First International Conference on Information Science and Engineering*, pages 1386–1390, Dec 2009.

[10] R. Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal Processing Magazine*, 29(5):128–132, Sept 2012.

[11] S. Hua, M. Kapoor, and D. C. Anastasiu. Vehicle tracking and speed estimation from traffic videos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW'18. IEEE, July 2018.

[12] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition*, pages 2756–2759, Aug 2010.

[13] Z. Kalal, K. Mikolajczyk, J. Matas, et al. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409, 2012.

[14] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[15] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[16] M. Naphade, D. C. Anastasiu, A. Sharma, V. Jagrlamudi, H. Jeon, K. Liu, M. C. Chang, S. Lyu, and Z. Gao. The nvidia ai city challenge. In *2017 IEEE SmartWorld Conference*, SmartWorld'17, Piscataway, NJ, USA, 2017. IEEE.

[17] M. Naphade, M. C. Chang, A. Sharma, D. C. Anastasiu, V. Jagarlamudi, P. Chakraborty, T. Huang, S. Wang, M. Y. Liu, R. Chellappa, J. N. Hwang, and S. Lyu. The 2018 nvidia ai city challenge. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW'18. IEEE, July 2018.

[18] H. Orlande, M. Colaço, G. Dulikravich, F. Vianna, W. da Silva, H. da Fonseca, and O. Fudym. Tutorial 10 kalman and particle filters. *Advanced Spring School: Thermal Measurements and Inverse Techniques*, 5:1–39, 2011.

[19] Y. Pei, S. Biswas, D. S. Fussell, and K. Pingali. An elementary introduction to kalman filtering. *CoRR*, abs/1710.04055, 2017.

[20] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[21] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[22] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *arXiv CoRR*, abs/1511.04136, 2015.

[23] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4705–4713, Dec 2015.

[24] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.