Fall 2018

Vehicle Tracking Based on Historical Intersection Over Union

Shuai Hua
San Jose State University

VEHICLE TRACKING BASED ON HISTORICAL INTERSECTION OVER UNION

A Thesis

Presented to

The Faculty of the Department of Computer Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Shuai Hua

December 2019

The Designated Thesis Committee Approves the Thesis Titled

VEHICLE TRACKING BASED ON HISTORICAL INTERSECTION OVER UNION

by

Shuai Hua

APPROVED FOR THE DEPARTMENT OF COMPUTER ENGINEERING

SAN JOSÉ STATE UNIVERSITY

December 2019

David C. Anastasiu, Ph.D.	Department of Computer Engineering
Mahima Agumbe Suresh, Ph.D.	Department of Computer Engineering
Gokay Saldamli, Ph.D.	Department of Computer Engineering

ABSTRACT

VEHICLE TRACKING BASED ON HISTORICAL INTERSECTION OVER UNION

by Shuai Hua

Multi-object tracking (MOT) could be applied to many video analysis scenarios, such as vehicle speed estimation, vehicle re-identification, and vehicle abnormal behavior detection. A tracking task can be formulated as a data association problem, for which there exist many different types of solutions. Track-by-detection is one of the most common approaches for the MOT task. In this paradigm, the tracking algorithm relies on the detection results to decide whether detected vehicles in sequential frames belong to the same track. In our work, we developed a reliable vehicle tracker following this paradigm, while considering the balance between tracking efficiency and tracking performance. Our algorithm extends the existing intersection over union (IOU) tracker and improves upon it by fusing historical tracking information. In addition, our tracker allows tuning certain hyperparameters that lead to improved results, including the minimum confidence score, the maximum confidence score, the IOU threshold, and the length of a candidate track. We demonstrated the effectiveness and efficiency of our approach using the UA-DETRAC benchmark dataset. Our proposed approach runs at an average speed of 28 frames per second (fps), which is 16 faster than one of the baselines but 24 times slower than the other. With regard to effectiveness, however, our approach outperforms both baseline methods by more than 20% in most of the tracking performance metrics and achieves a 60% performance improvement in certain cases. We conclude that our tracker, which balances running speed and performance, could be useful for applications running in a real-time environment.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor, professor David C. Anastasiu of the Computer Engineering Department at San José State University. I greatly appreciate his guidance in those times when I met difficulties with my research and writing. He always had very good answers for my questions.

I would also like to thank professor Mahima Agumbe Suresh and professor Gokay Saldamli who participated in my thesis defense committee. I am grateful for their valuable comments on my thesis. Without their passion and expertise, this thesis could not have been successfully finished.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Image processing and data analysis technologies emerged and became mature many decades ago. However, it is rare to see those two approaches used together to solve practical problems, usually due to the lack of data, which greatly affects the final results. The inadequacy of computing ability has also played a critical role in limiting the application of this technology to practical problems. The maturity of the Internet and hardware breakthroughs have recently made it possible to apply machine-learning technology to industry applications. These advancements have brought researchers and industry together to reconsider how to utilize existing data and develop novel algorithms that can train a machine to think and act like a human being.

The first priority in training a machine to act like a human is to teach the machine to recognize as many objects as possible. Recently, object detection has become one of the most popular domains in machine learning. Many researchers have achieved success by using different 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 follow (i.e., track) moving objects, including vehicles and pedestrians. There are many practical scenarios to which vehicle tracking can be applied. Autonomous driving systems, for instance, require the vehicle to have the capability to identify and track objects. Traffic surveillance systems also 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. In recent years, researchers have also tried to improve the accuracy of object detectors with the help of tracking 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 that 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 to stay unchanged for the same vehicle, from the first frame in which it emerges in the video to the end of the last, 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 usually becomes unstable when 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 the 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 weather conditions.

Many classical approaches used for tracking, such as particle filters [1], Kalman filters [2]–[5], and Bayesian filters [6], [7], have had some success in solving the problem. Those formulations belong to track-by-estimation methodology. However, in recent years, deep learning methods have been shown to greatly outperform these classic approaches. The success of deep convolutional neural network (DCNN) methods in the vehicle detection task makes it possible to build an object tracking approach on top of the object detector. This methodology is also known as *track-by-detection*. The track-by-detection pipeline works as follows: the detector first predicts the location of each object, represented as an in-frame bounding box and the confidence score of the detection; then, the tracker uses the detection results as input and generates the associated tracklets. Following this approach, the main task of a tracking algorithm is to consistently associate the bounding box with the correct ID for each vehicle across frames.

In reality, the detector does not always provide completely correct predictions, especially in situations when the vehicle becomes (partially) occluded by other vehicles or objects in the scene (e.g., light poles). A higher accuracy of the detector will lead to better tracking performance. On the other hand, a good tracker should be able to reduce the effects of interference in two ways. It should mitigate the effects (1) of false positive detections, i.e., objects reported by the detector as being vehicles that are not actually vehicles, and (2) of false negative detections, i.e., vehicles that the detector failed to detect, which often happens due to occlusion scenarios.

It is insufficient to evaluate tracking performance solely by relying on detection metrics, since the performance of different trackers varies given the same detection result. The evaluation protocol should have the ability to judge the overall results of detection and tracking, and to indicate the weaknesses and strengths of a tracker. In our work, we used a state-of-the-art tracking evaluation protocol designed for the UA-DETRAC tracking challenge [8]. This protocol analyzes tracking performance by combining both the detection and tracking performance results. We compared the performance of our tracker against two baseline methods, the intersection over union (IOU) tracker described by Bochinski et al. [9], a naïve vehicle tracker which only considers the overlap of localization bounding boxes between the current and previous frames in the video signal, and a tracker based on the Markov decision process (MDP), the MDP tracker, created by Xiang et al. [10]. While the algorithms behind these two trackers are different, both of them highly rely on the detection, since they are implemented following the track-by-detection paradigm. Our research contributions in this work include,

- we trained the well-known YOLO [11] object detector as a general vehicle detector and measure its effectiveness on the UA-DETRAC dataset,
- we evaluated the tracking efficiency and effectiveness of two recent tracking algorithms on the dataset – the IOU [9] and the MDP [10] trackers,
- we developed a novel tracker that extends the IOU tracker by taking into consideration historical IOU data, and
- we show that our tracker achieves a good balance of tracking effectiveness and efficiency performance compared to the baselines.

2 LITERATURE REVIEW

In the computer vision domain, the task of object detection and tracking has become very popular in recent years, in part due to recent advances in computation speeds and a boost in availability of quality data. Many researchers have published their achievements in this area, and many engineers in industry have applied these algorithms to solve practical problems.

The problem of vehicle tracking is usually formulated as follows: vehicle tracking is initiated by finding the location of and assigning an identifier (ID) to every vehicle in the first frame; then, the method must find the locations of each vehicle in the following consecutive frames, adding new vehicles to the list of tracked vehicles as they enter the frame. In this section, we are going to focus on reviewing some prediction-correction and track-by-detection approaches. Although there are some differences between these methods, the idea behind the two paradigms is similar: they both treat the tracking problem as a data fusion problem.

2.1 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 [1]–[6]. 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. [3] and Faragher [4]. The former presents an in-depth explanation of Kalman filters, while the latter provides an example of the Kalman filter-based movement model, which helps the reader to understand how this algorithm can be used in our problem domain. Pei et al. gave a straightforward definition of Kalman filtering: it is an algorithm that combines two imprecise estimations together, 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. In order to help the reader understand this abstract concept, Pei et al. presented the analogy of purchasing an estate, and then further explained this example from a statistical viewpoint. The authors of this paper successfully answered the question of how to fine-tune the unknown parameter in order to optimize the linear equation. In the second part of this paper, Pei et al. reviewed statistical concepts such as the mean, variance (and co-variance), and the Gaussian distribution. The authors initially explained these concepts with a simple scalar format, and then extended to a slightly more complicated vector and matrix format. 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 third and fourth sections of the article, the authors posed a good question about what optimize means in the framework of this problem, and then finally gave the detailed solution. Based on this knowledge, the authors then showed the reader how Kalman filters can be applied in a practical application example. We are thus able to understand that the prediction task is formulated as a linear equation, in which the measurement is injected to help correct the prediction, and the Kalman gain is used to represent the optimized coefficient of this linear fusion.

In the second article [4], Faragher presented the reader with an example that demonstrates the simple and intuitive idea behind deriving the the Kalman filter. Faragher's publication is especially targeted for the reader who does not have a strong mathematical background. Similarly to the article of Pei et al., Faragher also formulated the Kalman filters as a data fusion problem and optimized this problem by using the Gaussian distribution. But, unlike other articles, 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.

2.2 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. Farhadi and Redmon [11] improved upon an earlier state-of-the-art detector, called you only look once (YOLO). They obtained a faster and more accurate detector to generate decent boxes for instances by using multi-scale predictions without specifying the anchor boxes. Fergus and Zeiler [12] 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 [13].

Object tracking research has also gained popularity recently. Bochinski et al. published an intuitive idea for the vehicle tracking problem, known as the IOU tracker [9]. There have been two achievements that merit mentioning in this publication. Firstly, this model outperforms others with regards to efficiency, resulting in very high vehicle tracking speeds, in the order of 100,000 frames per second (fps). Secondly, the algorithm is implemented without considering image information, making it easy to adopt by the novice who has little computer vision knowledge. 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. This thesis will use this algorithm as one of the comparison baselines and we describe the method in detail in Section 4.

Bewley et al. implemented a simple online and real-time tracking algorithm based on the IOU tracker, called SORT [14]. 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. Bewley et al. paid more attention to developing a fast tracker, without considering common tracking issues such as occlusions. In their proposal, SORT formulates the object movement as a linear model, similar to the Kalman filter models, and predicts the location in the new frame based on the position estimate of the model. This paper uses the FrRCNN [15] detector. Bewley et al. used the IOU distance to fuse the prediction and measurement results. The authors computed the optimal model parameters using the Hungarian algorithm [16] developed by Kuhn. Moreover, they note that their method is able to implicitly solve some of the short-term occlusion problems because the tracker will track occluded objects as well.

Xiang et al. formulated the tracking problem as a decision making problem [10]. The authors relied on the Markov decision process to solve the problem. Hence, they named their method the MDP tracker. This thesis will also use this algorithm as one of our baselines and we describe the method in detail in Section 5.

Kalal et al. proposed a tracking algorithm called median flow (MF) [17]. The authors used the forward-backward error (FB error) algorithm to detect the failure in the tracking task. The FB error approach is based on the assumption that two tracking results should be very close to each other no matter whether the tracking starts from the first frame or from the opposite direction. The terms forward tracking and backward tracking are used to indicate the starting point. The former means that tracking happens starting with the first frame, while the latter indicates that tracking begins from the last frame and happens in reverse. For instance, a tracker can obtain two similar trajectories by starting from two opposite points. If a significant difference exists between these two trajectories, then the forward trajectory should be marked as an error. In this paper, the authors first explained the general idea and gave the mathematical derivation for the FB error. Then, the authors started to demonstrate the idea by tracking a single point and then extended 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 [18]. 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.

Kalal et al. also developed a novel *tracking-learning-detection* framework on the detection and tracking task, known as TLD [19]. As indicated in 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. The authors used two "experts" to help detect and solve the false positive and the false negative errors. The P-expert is used to identify the false negative error; while the N-expert is responsible for observing the false positive error. In this paper, 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 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 [20] presented a metric to evaluate object detection and tracking systems in early 2006. Bernardin and Stiefelhagen [21] 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 in their publication [8], which they called the UA-DETRAC MOT benchmark. We evaluated the performance of the baseline trackers and our method by using this protocol.

2.3 The UA-DETRAC Benchmark Dataset

An MOT benchmark should consist of an annotated dataset as well as the evaluation metric to verify the performance of the entire MOT system. In this section, we focus on reviewing the UA-DETRAC benchmark 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 [8], and they also introduced a new protocol to evaluate the MOT system, which they called the UA-DETRAC MOT protocol. They provided an implementation of the protocol in Matlab and C++, which currently can be run under the Windows and Linux platforms. The dataset they provided 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. In the UA-DETRAC dataset, the authors labeled four types of objects, including car, bus, van, and other, where “other” represents some low-resolution regions existing in 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 the snapshot of the UA-DETRAC benchmark in Fig. 1.

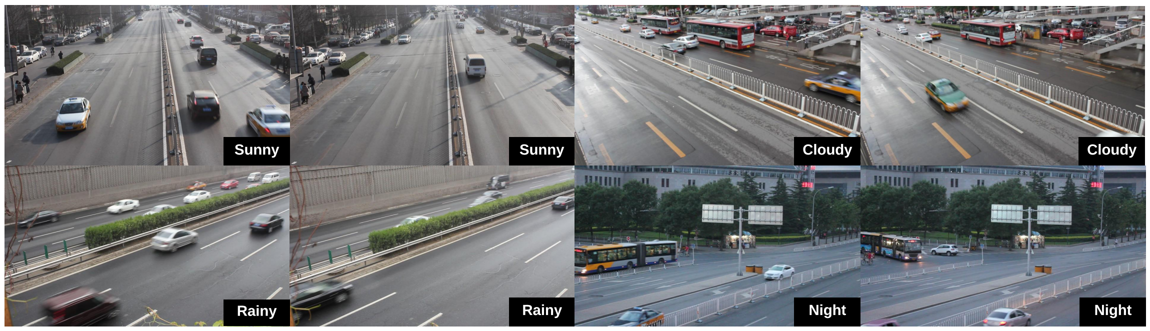

Fig. 1. The UA-DETRAC Benchmark. An example of the dataset under different conditions.

Similar to other tracking datasets, the UA-DETRAC dataset is divided into a training and a test set with a ratio of 6:4. The training videos are captured at different locations and under different traffic conditions, and the authors ensured that the test set has a similar distribution of traffic conditions as the training set.

2.4 The UA-DETRAC Evaluation Protocol

Despite the significant progress that has already been achieved in object detection and tracking, researchers must still rely on the classical approach to evaluate the system, in which the detector and the tracker is judged separately. For the interested reader, traditional evaluation approaches are summarized well by Bashir and Porikli [20]. The most widely accepted method was to evaluate different trackers under the same detector. Ideally, a perfect metric to rank a tracker should be a score that represents the comprehensive performance of the tracking algorithm. Comprehensive here means considering the accuracy of the detector along with the accuracy of the tracker. 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 [8]. This has been widely known as the UA-DETRAC benchmark. The authors not only contributed the annotated benchmark already mentioned, but also provided an MOT system ranking protocol which has demonstrated significant impact. In the following section, we first review the metrics for detection and tracking separately, then learn how to generate the overall rank score by combining the two metrics.

2.4.1 Detector Evaluation

The UA-DETRAC protocol uses the precision versus recall (PR) curve to learn the performance of the detector. The PR curve is created by measuring precision and recall at increasing levels of detection confidence. 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.

2.4.2 Tracker Evaluation

There are different indicators describing the performance of the tracker from variable viewpoints. We will explain the details of each metric in this section.

- Mostly Track. 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. Obviously, the greater the MT, the better the tracker. In our work, we set $m = 80$.
- Mostly Lost. 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. Ideally, the tracker will assign a unique ID to each object, and this ID will be kept unchanged across the entire trajectory of the object. However, in some cases, such as due to occlusion, it is possible that the tracker will assign a different ID to the same object after some time. Identity switches (IDS) is a number used to count the ID changes. A perfect tracker should have IDS=0.
- Fragmentation of the Trajectory. During tracking, fragmentation (FM) will happen if the tracked object disappears in some frames and re-appears again, resulting in the discontinuity of the trajectory. The FM score is defined as the percent of tracks that were fragmented.
- False Positive. Considering the example of a car detector, the false positive (FP) rate indicates the percentage of cars that the detector fails to correctly detect across all frames.
- False Negative. Continuing our car detector example, the false negative (FN) rate describes the situation in which the detector makes errors in predicting other objects as cars.

- Multi-Object Tracking Accuracy. Multi-object tracking accuracy (MOTA) of a certain sequence is defined as

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

  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

$$MOTA = 100 \times \left(1 - \frac{\sum_v \sum_t (FN_{v,t} + FP_{v,t} + IDS_{v,t})}{\sum_v \sum_t GT_{v,t}}\right), \tag{2}$$

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

- Multi-Object Tracking Precision. Multi-object tracking precision (MOTP) is defined as

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}, \tag{3}$$

  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.

- False Alarm Rate. False alarm rate (FAR) is defined as

$$FAR = \frac{FP}{TP + FP}. \tag{4}$$

2.4.3 *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, the PR-FM score, the PR-FP score, and the PR-FN 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. 2. 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.

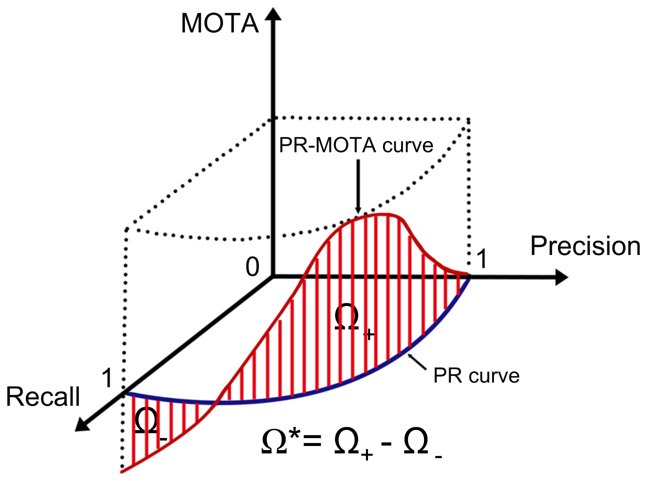

Fig. 2. PR-MOTA curve. Ω^* is a value describing the overall performance of the MOT system.

3 VEHICLE DETECTION BY YOLO

We 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 [11] as our vehicle detector, which we will introduce in this section.

3.1 Training Data and Test Data

As mentioned in Section 2.3, we used the UA-DETRAC dataset [8] for training and testing. The UA-DETRAC benchmark mainly targets multi-vehicle detection and tracking. There are a total of 100 sequences in this benchmark, and the number of training sequences and test sequences are 60 and 40, respectively. The dataset is split into 3 different levels based on the weather condition (sunny, cloudy, rainy, and night) in which those videos have been recorded. The weather condition mainly affects the illumination of the scene. An image taken on sunny days should be clear and much easier to be analyzed, while night-time images may be affected by blur, which may reduce the performance of the detector. Considering that the annotated test sequences are not publicly available, we can only rely on the training sequences for our training and evaluation purposes.

In our experiment, we used the annotated UA-DETRAC training set as input for the YOLO detector, which we then split into training and test sets with a ratio of 8:2 samples. Both the baseline trackers and our tracker used the same detection results as input for predicting tracks. 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. Since the IOU-tracker and our tracker only depend on the detection result, we could apply the remaining 12 sequences to these two trackers. Unfortunately, there are some frames missing in 3 of these 12 sequences. Therefore, while we used these 12 sequences to evaluate the performance of the detector, we ignored the 3 sequences with missing data (39781, 40152, and 63544) from our tracking test dataset.

As Xiang et al. suggested [10], the MDP tracker depends not only on the detection output, but also on the dataset to train the tracker; consequently, we had to reserve nearly half of the test data (4 sequences) to train the MDP tracker and the remaining 4 sequences were used to evaluate the tracker. Considering the above factors, the final test set used to compare the two baseline trackers and our tracker contained 4 sequences.

The training sequences used for the MDP tracker are listed in Table 1 and the test sequences used for all trackers are listed in Table 2. The complete list of training sequences used in our experiments are available in Appendix A. For all sequences, we also provide summary statistics, including number of frames, number of bounding boxes, number of vehicles in each category (cars, vans, buses, and other), and the weather conditions when the video was recorded.

Table 1
Training Sequences for MDP Tracker

Sequences	Frames	Boxes	Cars	Vans	Buses	Other	Weather	Track length
20011	664	7,655	7,053	296	95	211	sunny	53
39771	570	3,605	2,933	161	511	0	night	27
40141	1,600	6,222	4,916	1,306	0	0	cloudy	33
63521	2,055	15,088	11,806	962	2,099	221	rainy	90
Total	**4,889**	**32,570**	**26,708**	**2,725**	**2,705**	**432**	-	**203**

Table 2
Test Sequences for HIOU, IOU, and MDP Tracker

Sequences	Frames	Boxes	Cars	Vans	Buses	Other	Weather	Track length
20012	936	8,608	6,481	790	1,337	0	sunny	42
39801	885	4,853	4,828	0	0	25	night	43
40131	1,645	15,324	12,117	969	2,238	0	cloudy	67
63525	985	3,470	2,097	212	1,161	0	rainy	32
Total	**4,451**	**32,255**	**25,523**	**1,971**	**4,736**	**25**	-	**184**

3.2 Training Data Format

It is necessary to provide the dataset and the label when training a model with a supervised learning approach. While training our detection model, we simply put all training data into one folder and renamed all image files with their sequence name plus a

5-digit frame number (e.g., 40201_img00803.jpg), indexing from 1. For each image, we created one text file with the same name but ".txt" extension which contained ground truth annotations. An example of such a ground truth file is given in the Table 3. 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.

Table 3
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

3.3 Training YOLO

We trained our YOLO detection model using an NVIDIA Titan Xp GPU. We used the darknet53.conv.74 model weights as an initial starting point for our model training. Training time took 50 hours. We set the maximum number of iterations to 50,000 and saved the model weights every 10,000 iterations. We used the final model, saved after 50,000 training iterations, in our experimental evaluation.

3.4 Pre-processing the Detection Result

For each input image, the prediction output is a text file in JSON format containing the confidence score, the object class and its bounding box. When setting the confidence threshold to 0.0, the size of the prediction output is approximately 42 GB. We found that this result includes an excessive number of false alarms, which in turn have very small confidence scores and significantly slow down tracking. We chose to reduce the false positive results, by ignoring those results with confidence below a minimum confidence level, 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.

3.5 Preparing for Precision and Recall

As shown in Fig. 2, the UA-DETRAC MOT metric is highly dependent on the precision-recall curve to evaluate the performance of the tracker. We generated the precision-recall curve by evaluating the training sequences, test sequences, and track sequences. We iterated all sequences for different thresholds, and counted the true positive and false positive detections by comparing the prediction bounding box with the ground truth bounding box. According to the UA-DETRAC MOT metric, we set the IOU between the prediction and ground truth bounding box to 0.7. We plotted the precision-recall curve to help us understand the performance of the detector. We generated this 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. 3 shows the precision-recall curves among the training sequences (left), test sequences (middle), and track sequences (right). 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.

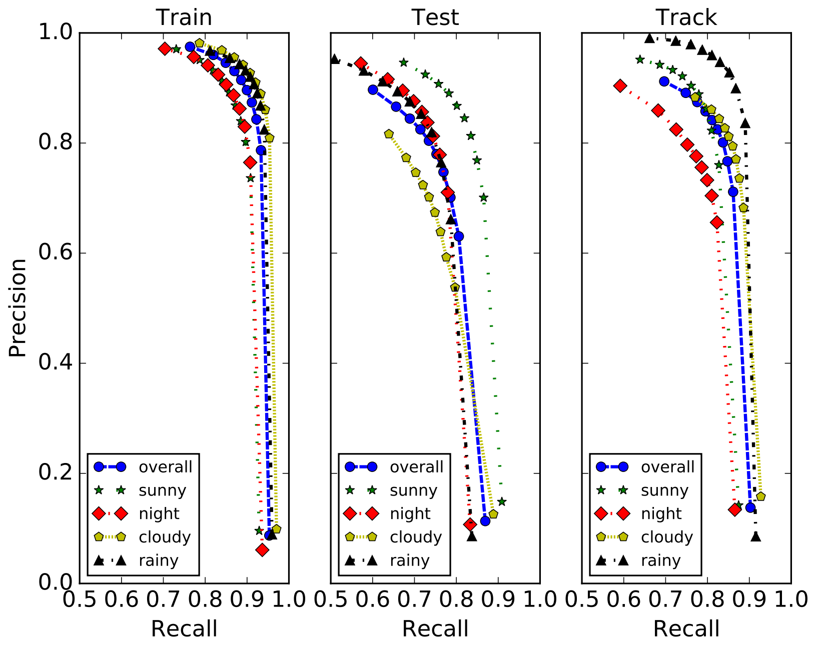

Fig. 3. Precision-recall curve under different weather conditions.

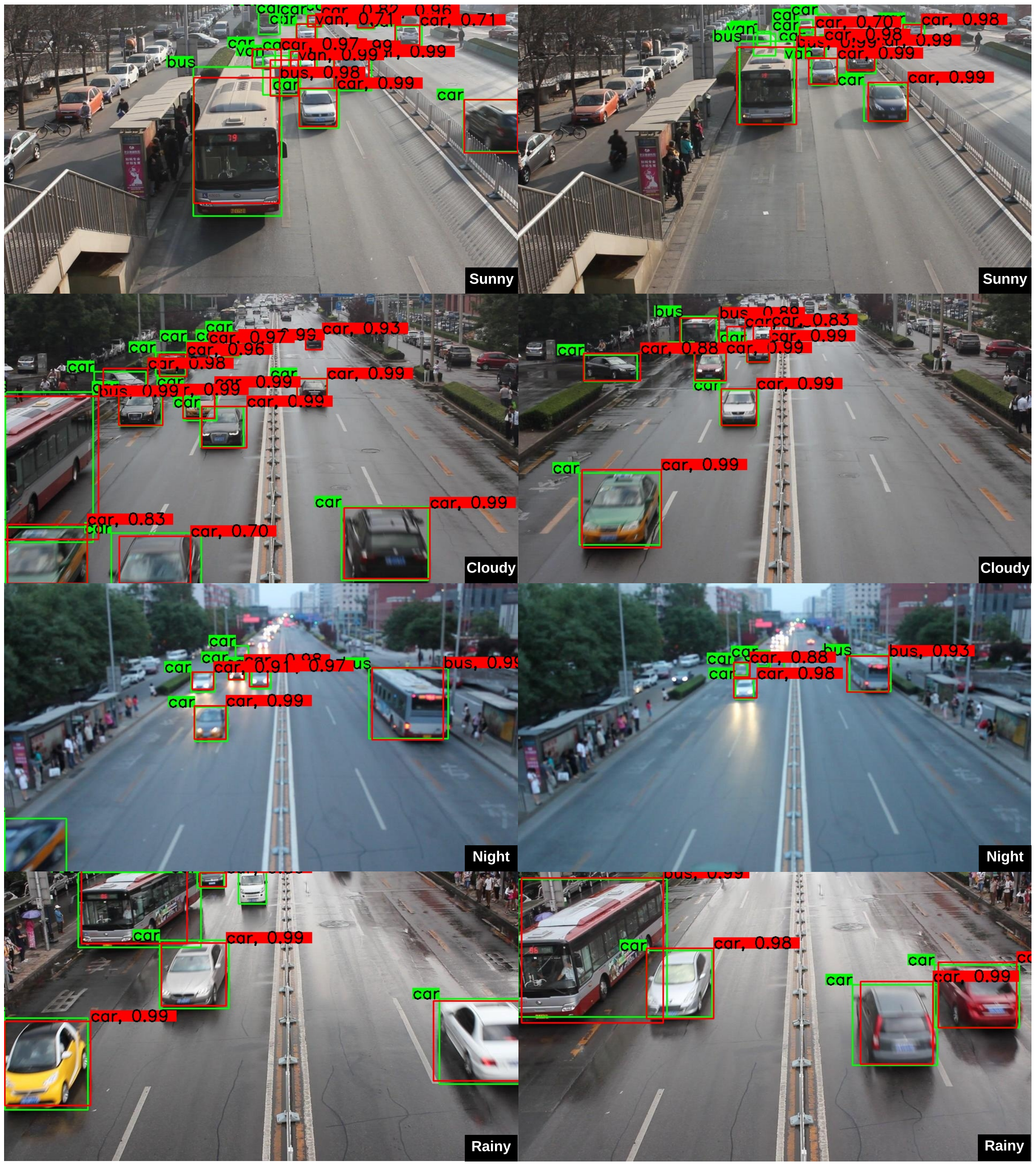

Fig. 4. An example of the visualization of the YOLO detection results versus ground truth annotation under different weather conditions.

4 VEHICLE TRACKING USING THE IOU TRACKER

As its name (intersection over union) indicates, Bochinski et al. [9] 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

$$IOU(a,b) = \frac{Area(a) \cap Area(b)}{Area(a) \cup Area(b)}. \tag{5}$$

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 by its simplicity, resulting in very high tracking speeds.

4.1 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.

4.2 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.

4.3 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.

4.4 IOU Between Two Consecutive Frames

As the authors introduced in the paper "High-Speed Tracking-by-Detection Without Using Image Information" [9], 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.

5 VEHICLE TRACKING BY MARKOV DECISION PROCESS

Xiang et al. formulated the MOT task as a decision making problem in their paper "Learning to Track: Online Multi-Object Tracking by Decision Making" [10] and used the Markov Decision Processes (MDPs) to solve this problem. The authors associated an object with different states in its lifetime and linked a certain *policy* in each state so that the model could transition from one state to another. One of the targets of this model is to learn the polices for all possible state transitions. The authors partitioned the lifetime of an object into four states, which they named "active," "tracked," "inactive," and "lost." The active state is the initial state of each object. The algorithm initiates an object with the active state when it was first predicted by the detector. Depending on the confidence score, the method could transition the object either to the inactive state or to the tracked state. If the confidence score is greater than a certain threshold, it indicates that this is a true positive prediction, so the method assumes that this object should be tracked and sets the state of the object to tracked. Otherwise, this was a false alarm and the method marks the state of the object as inactive. Ideally, an object in a tracked state will continue to be tracked until it leaves the video frame or moves far enough away from the camera to no longer be detectable. However, occlusions are likely in the real world. Thus, if an object is no longer being detected and it did not exit the video frame, the method sets the state of the object as lost. If an object goes missing for a certain number of frames and then re-appears, it should be tracked again. In this case, the method transitions the object state back to tracked. Otherwise, if the object is in the lost state for too many frames, the method will terminate the tracking by changing its state to inactive. We illustrate this state transition process in Fig. 5.

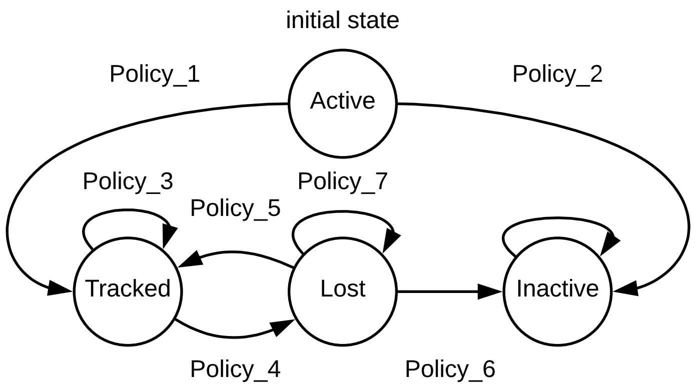

Fig. 5. The MDP state transition.

5.1 Policy in Active State

Xiang et al. developed two options when an object is in its active state. They may change it to a tracked state or move it to an inactive state. In this framework, Xiang et al. trained a binary support vector machine (SVM) [22] to help make the decision. The authors chose two types of datasets to train the binary classifier. They selected the noise detections for one dataset, in which the confidence score was smaller than a certain threshold, and they used the true positive prediction as another dataset. The authors normalized the values of the top-left, width, height, confidence score for the feature vector in an effort to improve classification accuracy.

5.2 Policy in Tracked State

Xiang et al. formulated the MDP tracker to have the ability to make decisions under the tracked state. The authors utilized the appearance model to help the MDP tracker make decisions with regards to changing the state of the object. By comparing the appearance similarity of targets between the new frame and the previous frame, as long as the target can be seen in a new frame, their method assigns this object to the tracked state. In this framework, the authors combined some relatively expensive metrics, such as the

FB error [17], with some light-weight features, such as bounding box overlap and confidence score, to build the appearance template.

5.3 Policy in Lost State

The authors assumed that, if an object was in the lost state, it could be switched to an inactive state, tracked state, or a lost state. Setting the object to an inactive state was based on the count of the number of frames that the object was not found in. If they found the object lost for some number of frames greater than a certain threshold, then they assumed the object to be in an inactive state. However, it is a little challenging to differentiate whether, when an object re-appears, it is indeed a new object or an old object recovering from a lost state. The authors viewed this problem as a data association problem and solved it using an SVM binary classifier. They trained the binary classifier to have the ability to generate a value indicating the probability that these two detections are linked, when providing the similarity of these two detections.

6 VEHICLE TRACKING BY HISTORY-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. [9] 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. We show two common interference conditions during tracking in Fig. 6.

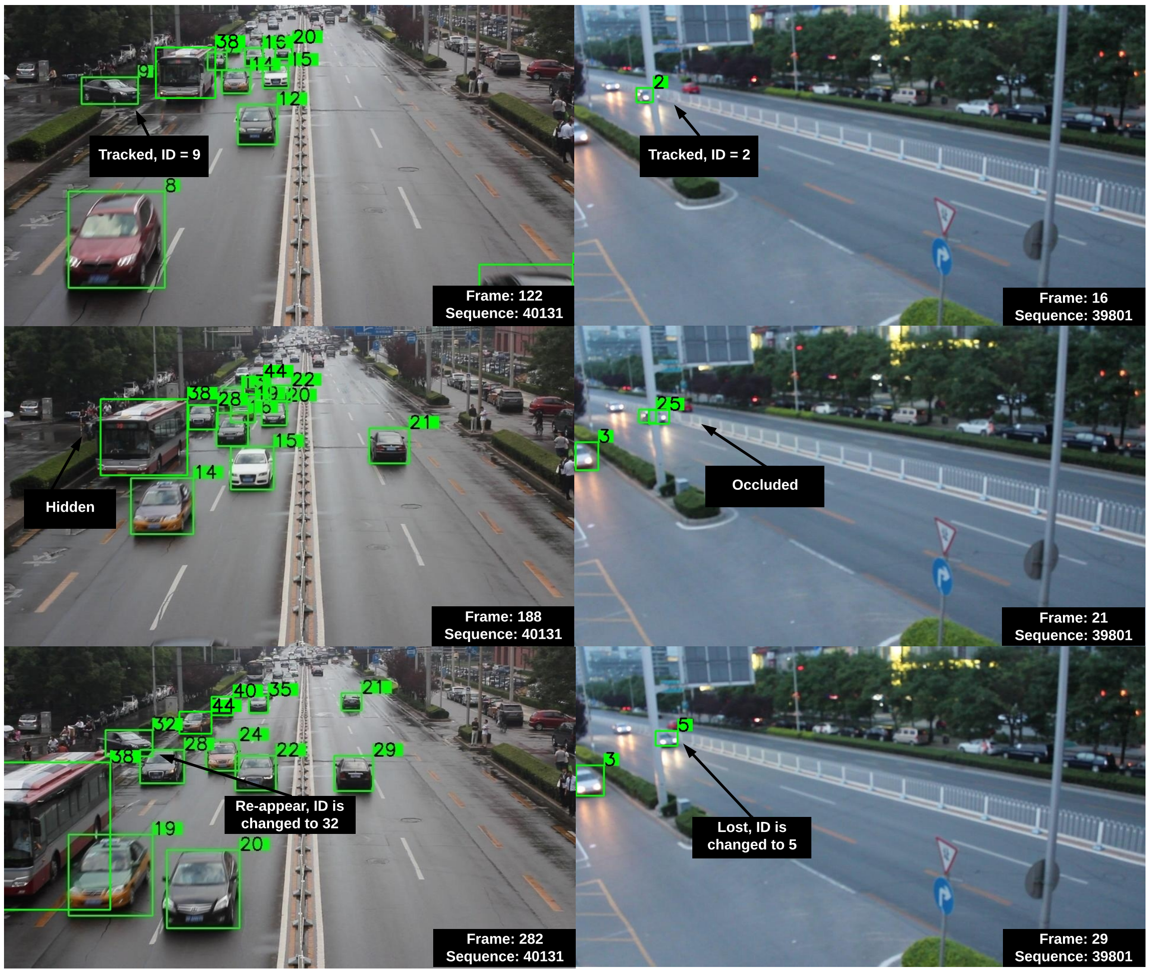

Fig. 6. Two common occlusions.

In Fig. 6, the left 3 images demonstrate the scenario in which a tracked vehicle (original ID is 9) is temporarily hidden by a bus later re-detected. We expect that the tracker is smart enough to assign the same ID to these two cars because they are in fact the same exact car. Unfortunately, both the IOU tracker and the tracker we implemented (HIOU) are unable to handle this type of noise, because they lack image information. Our HIOU tracker's advantages over the IOU tracker is in its ability to deal with the interference of non-vehicle occlusions. We show this situation in the right side of Fig. 6. In this case, although the detector made a mistake to predict one vehicle as two because of the occlusion (pole), our algorithm is still able to work correctly since our design considers tracking history.

We mainly use the IOU to associate the bounding boxes and do not introduce any image information in our proposal; therefore, we can also simplify the complicated vehicle tracking task to a bounding box association problem. We differentiate our algorithm with the IOU tracker by adding the tracking history. Similar with the IOU tracker, we only track those detections in which the confidence scores are above a certain threshold. We use this way to prevent adding false alarms to the tracker and increase tracking accuracy. For the IOU tracker, the author assigned a new ID to the detection *immediately* when they found 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 false detections, such as those shown in Fig. 6. 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:

$$\theta' = \theta - 0.1,\ where \min_{\theta} \geq 0.3. \tag{6}$$

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. We illustrated our algorithm in Algorithm 1 and implemented it in Python 3.5 without any special optimization. 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.

Algorithm 1 HIOU Tracker

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

We further improve the performance of our tracker with two more 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.

7 EXPERIMENTS AND RESULTS

The pipeline of our experiment is shown in Fig. 7. We used YOLO to localize vehicles in the UA-DETRAC benchmark dataset. We described the dataset partition and the training details in Section2.3. 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. It took our method approximately 50 hours to finish the training for the entire dataset. Additionally, the method spent about 50 hours generate the predictions for all 48 training sequences, and another 11 hours to generate vehicle detections for the 12 test sequences. All of these predictions were done using a confidence score threshold of 0.0.

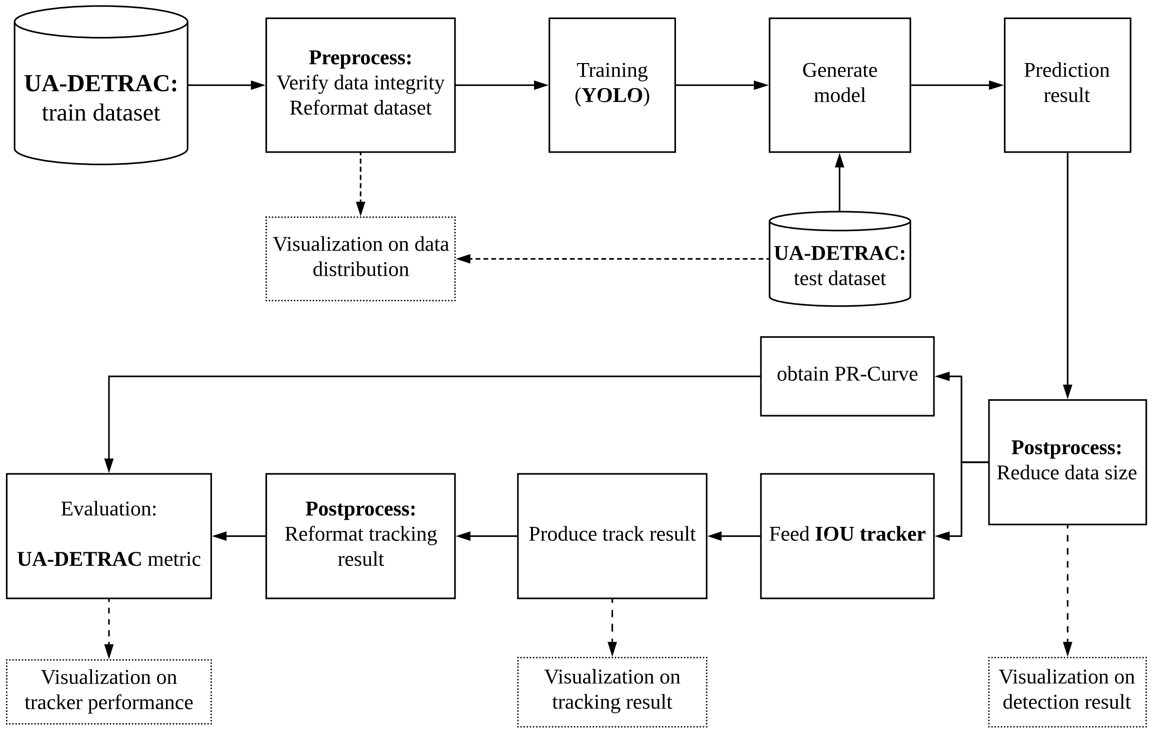
UA-DETRAC: train dataset
Preprocess: Verify data integrity Reformat dataset
Training (YOLO)
Generate model
Prediction result
Visualization on data distribution
UA-DETRAC: test dataset
obtain PR-Curve
Postprocess: Reduce data size
Evaluation: UA-DETRAC metric
Postprocess: Reformat tracking result
Produce track result
Feed IOU tracker
Visualization on tracker performance
Visualization on tracking result
Visualization on detection result

Fig. 7. Pipeline of IOU tracker by YOLO detection.

We used another server equipped with 2 Intel(R) Xeon(R) E5-2680 v3 CPUs and 384 GB memory to run all the tracking experiments for our method and the two baseline trackers. We compared the performance in all tracking experiments using the same video sequences and detection results. The test sequences we used in our experiments are listed in Table 2.

Both the IOU tracker and HIOU tracker are developed in Python, while the MDP tracker is implemented using Matlab. Note that the authors of the MDP tracker evaluated their algorithm using the MOT benchmark [23], which was introduced by Leal-Taixé et al. and mainly focuses on pedestrian tracking. However, we believe that this algorithm is able to be used for vehicle tracking as well.

We chose the UA-DETRAC MOT evaluation protocol to evaluate the tracking performance among these three trackers. In order to learn how the hyper-parameters affect the performance of the 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 listed the range of the parameters in Table 4. In this table, we put a dash in the cell to indicate that this parameter is not applicable for this tracker. We show a typical tracking result for the HIOU tracker, the IOU tracker, and the MDP tracker in Fig. 8. Fig. 9 gives an example of the capability of the 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).

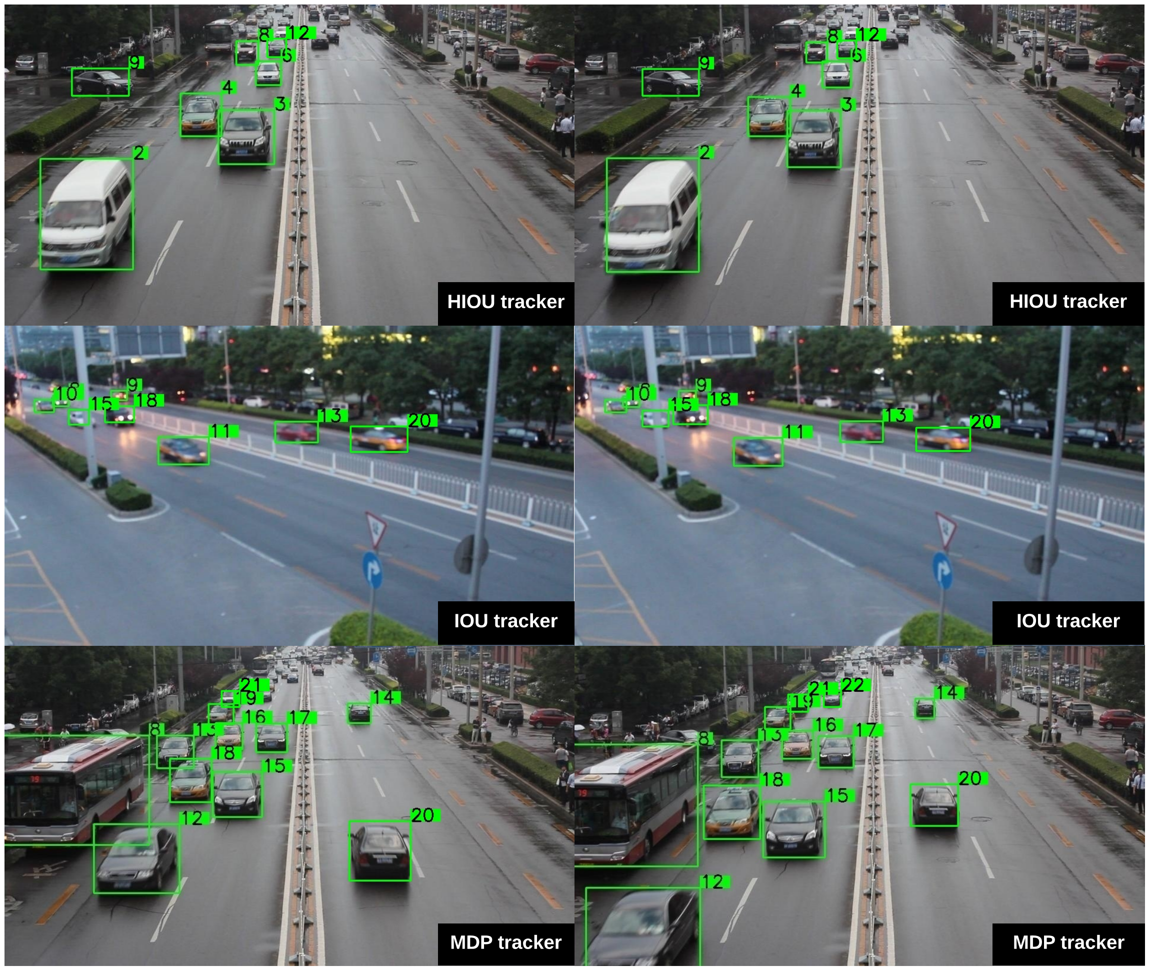

Fig. 8. Visualization of the tracking results among three trackers.

Table 4
Range of the hyper-parameters Among HIOU, IOU, and MDP Tracker

Tracker	IOU	Max score	Min score	Track length	Historical frame
HIOU	0.3 - 0.8	0.5 - 0.9	0.0 - 0.9	2	3
IOU	0.3 - 0.8	0.5 - 0.9	0.0 - 0.9	2	-
MDP	0.3 - 0.8	-	0.0 - 0.9	-	-

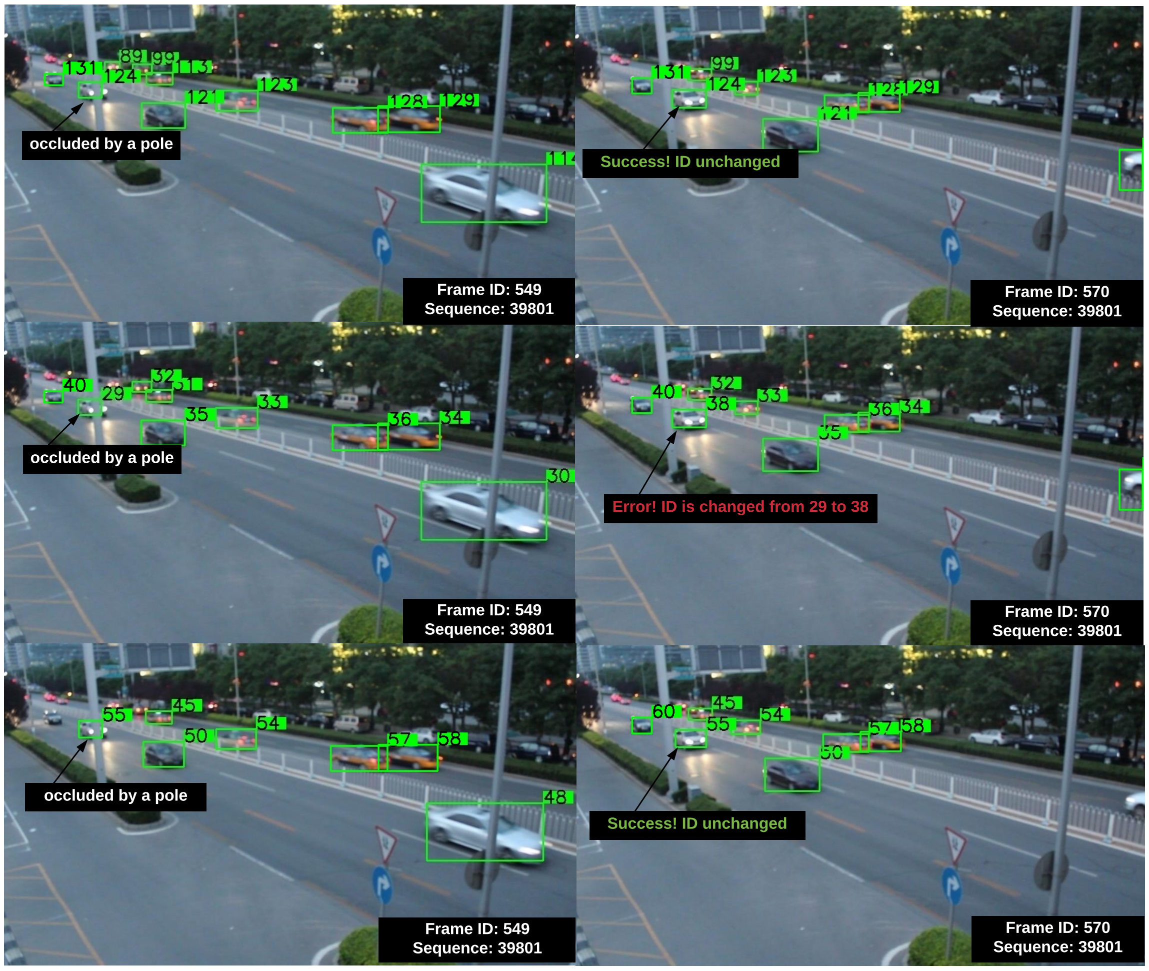

Fig. 9. The capability of trackers to overcome the non-vehicle occlusion (a pole).

In the rest of this section, we will describe two types of evaluation results. First, we present results from joining the detector and the tracker together. Second, we list the tracking results alone without considering the performance of the detector. In order to obtain comprehensive 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 values by applying different hyper-parameters, and we selected the best values for each method, which we include in Table 5. The values of the hyper-parameters that lead to the best scores are shown in Appendix B.

Table 5
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

We also used the metrics MT, MOTA, MOTP, IDS, FM, FAR, and ML without considering the performance of the detector. We visualized the relationship between the confidence score and the metric MT, MOTA and MOTP in Fig. 10. Additional results for other metrics are included in Appendix C. 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, the IOU tracker, and the MDP tracker. In this experiment, we only varied the IOU threshold from 0.5 to 0.8. Moreover, we were also interested in how the IOU threshold affects these tracking metrics. We analyzed this relationship in Fig. 11. We note that the IOU threshold affects result quality very little in the MDP tracker, which is reasonable since the contribution of the overlap in this algorithm is not too high; however, the overlap plays an important role in both the HIOU tracker and IOU tracker. One should note that setting the IOU greater than 0.6 will drop the most track (MT) performance.

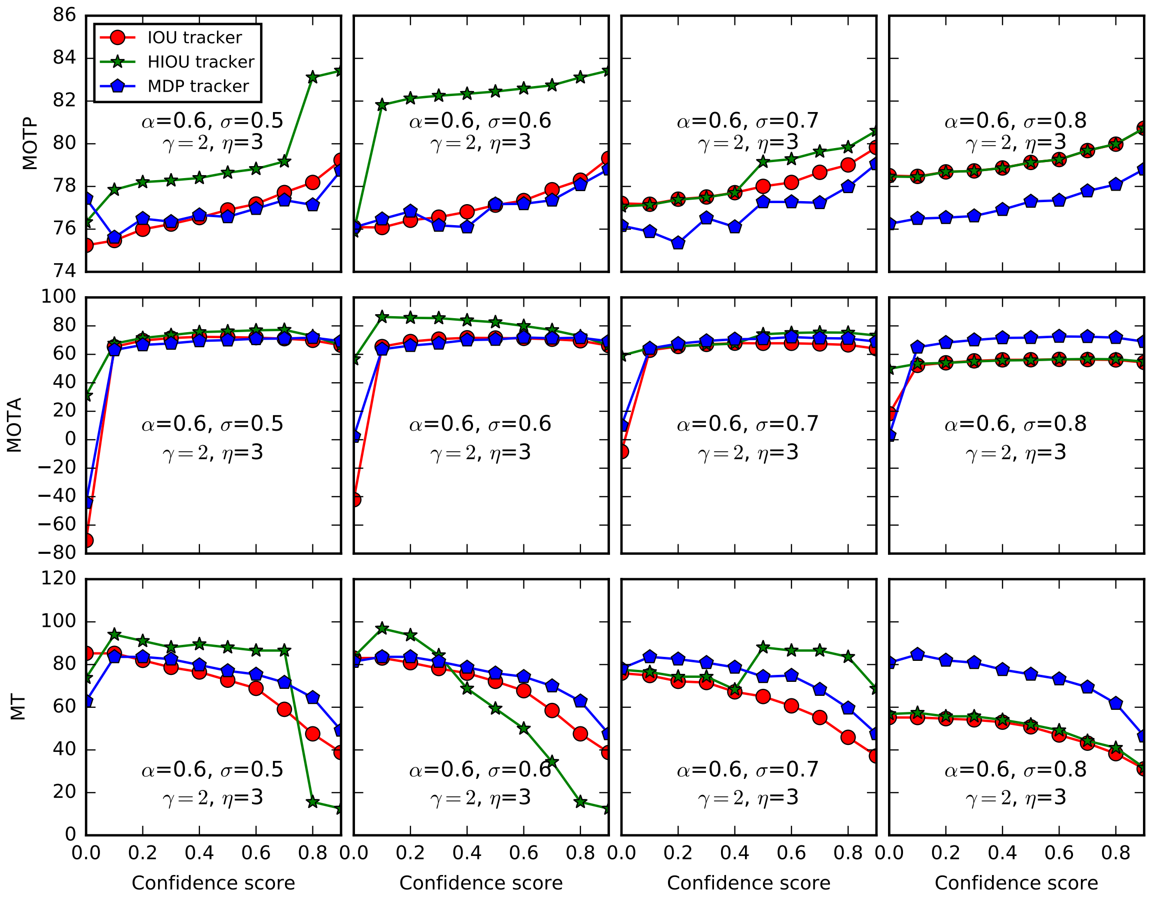

Fig. 10. The overall performance in 4 sequences among the HIOU tracker, the IOU tracker, and the MDP tracker.

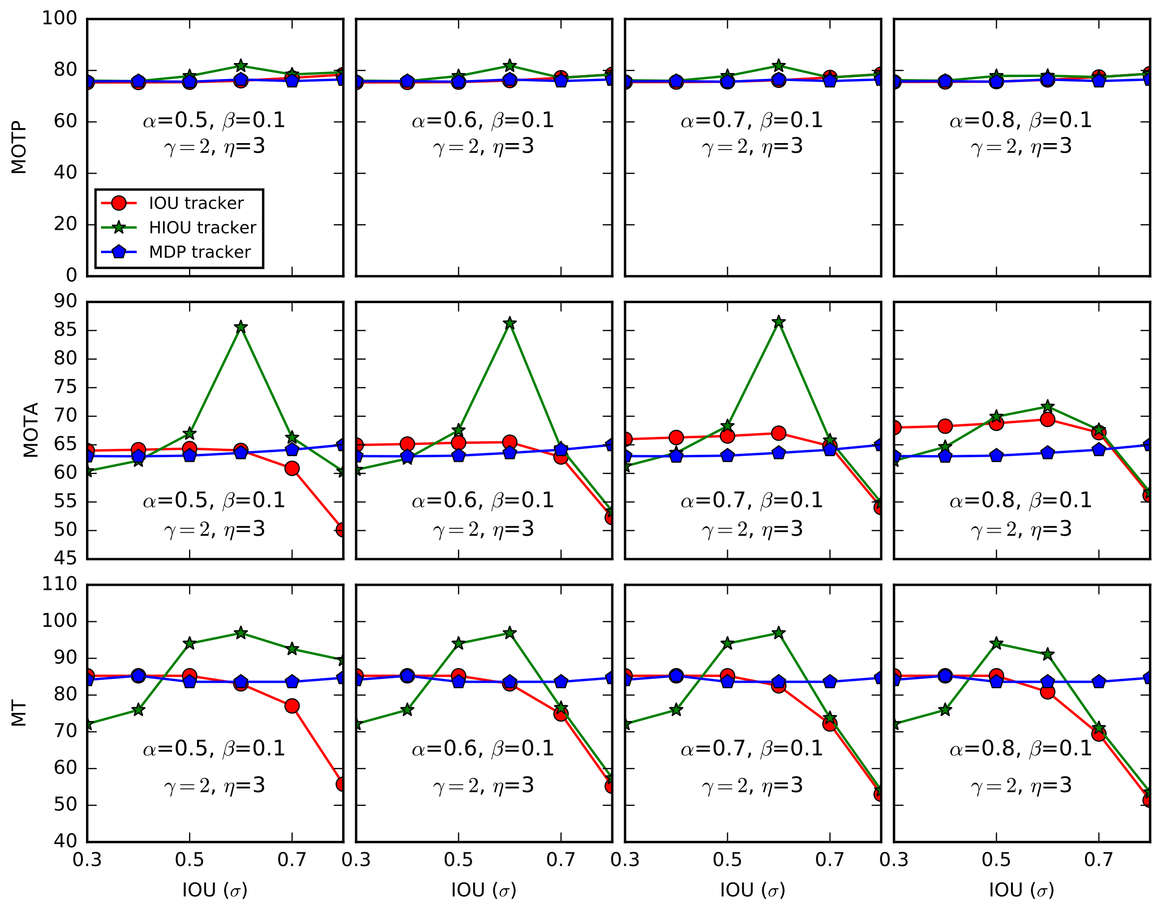

Fig. 11. The overall performance in 4 sequences among the HIOU tracker, the IOU tracker, and the MDP tracker.

The running speed is one of the most critical metrics to evaluate a tracker. We have to balance the 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 two baseline trackers and show these results in Table 6. Results show that the IOU tracker outperforms our tracker and the MDP tracker with regards to effectiveness; however, our tracker is able to greatly outperform the MDP tracker with regards to efficiency.

Table 6
Running Speed (fps) of HIOU, IOU, and MDP Tracker

Trackers	**20012**	**39801**	**40131**	**63525**
Confidence score 0.0				
HIOU	4.23	11.34	2.99	8.96
IOU	**40.50**	**87.15**	**43.63**	**62.00**
MDP	0.37	0.48	0.41	0.69
Confidence score 0.1				
HIOU	557.73	950.40	347.85	2,382.62
IOU	**1,647.13**	**2,280.29**	**1,512.48**	**3,336.67**
MDP	2.66	3.26	2.39	7.31
Confidence score 0.2				
HIOU	736.20	1,133.34	491.19	2,838.86
IOU	**1,793.85**	**2,404.67**	**1,672.43**	**3,496.80**
MDP	2.50	3.46	2.46	6.86
Confidence score 0.3				
HIOU	857.64	1,324.41	665.77	3,066.27
IOU	**1,910.51**	**2,486.94**	**1,764.37**	**3,506.06**
MDP	2.74	3.59	2.59	7.40
Confidence score 0.4				
HIOU	1,061.79	1,414.15	760.07	3,211.34
IOU	**1,987.38**	**2,525.25**	**1,797.85**	**3,592.22**
MDP	2.94	3.58	2.58	7.42
Confidence score 0.5				
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
Confidence score 0.6				
HIOU	1,317.74	1,743.39	949.92	3,276.81
IOU	**2,136.77**	**2,701.84**	**1,906.53**	**3,705.25**
MDP	2.96	3.69	2.64	7.53
Confidence score 0.7				
HIOU	1,398.67	1,917.88	1,109.91	3,414.49
IOU	**2,187.37**	**2,758.00**	**1,990.79**	**3,782.72**
MDP	3.00	3.86	2.64	7.83
Confidence score 0.8				
HIOU	1,449.72	2,264.52	1,230.00	3,497.80
IOU	**2,244.21**	**2,877.62**	**2,093.79**	**3,842.30**
MDP	3.13	4.09	2.66	7.86
Confidence score 0.9				
HIOU	1,762.25	2,709.34	1,477.01	3,636.81
IOU	**2,446.64**	**3,154.73**	**2,246.79**	**3,936.80**
MDP	3.32	4.73	2.75	8.29
Average speed				
HIOU	40.81	106.21	28.88	87.32
IOU	**342.80**	**670.78**	**359.89**	**537.52**
MDP	1.71	2.22	1.69	3.77

8 CONCLUSIONS

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 evaluate our algorithm. We used the existing and well-known object detector YOLO to predict vehicles. We found that the overall speed of our tracker is higher than that of the MDP tracker, but lower compared to the IOU tracker. We obtained a relatively high PR-MOTA metric over these 4 sequences compared to the IOU tracker and the MDP tracker. It is significant to note that the overall ID switch score in our tracker is 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 objects.

9 FUTURE WORK

Our work mostly considered the trade-off between tracking efficiency and accuracy. We ultimately implemented an approach to solve the vehicle tracking task without adding any image information. We improved the existing IOU algorithm by adding the history tracking information, resulting in a tracker with better performance in speed and comparable tracking accuracy as the baselines. 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 solve such occlusion issues, and, at the same time, 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.

Literature Cited

[1] 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*, vol. 5, pp. 1–39, 2011.

[2] G. Bishop, G. Welch, *et al.*, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-3175, p. 59, 2001.

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

[4] R. Faragher, "Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]," *IEEE Signal Processing Magazine*, vol. 29, pp. 128–132, Sept 2012.

[5] 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*, pp. 1386–1390, Dec 2009.

[6] F. Dellaert and C. Thorpe, "Robust car tracking using kalman filtering and bayesian templates," in *Conference on Intelligent Transportation Systems*, vol. 1, 1997.

[7] B. Yang, C. Huang, and R. Nevatia, "Learning affinities and dependencies for multi-target tracking using a crf model," in *CVPR 2011*, pp. 1233–1240, June 2011.

[8] 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*, vol. abs/1511.04136, 2015.

[9] 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)*, pp. 1–6, Aug 2017.

[10] 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)*, pp. 4705–4713, Dec 2015.

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

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

[13] 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)*, pp. 3908–3916, June 2015.

[14] 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)*, pp. 3464–3468, 2016.

[15] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015.

[16] H. W. Kuhn, *The Hungarian Method for the Assignment Problem*, pp. 29–47. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

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

[18] 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, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[19] Z. Kalal, K. Mikolajczyk, J. Matas, *et al.*, “Tracking-learning-detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, p. 1409, 2012.

[20] F. Bashir and F. Porikli, “Performance evaluation of object detection and tracking systems,” in *Proceedings 9th IEEE International Workshop on PETS*, pp. 7–14, 2006.

[21] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: The clear mot metrics,” *EURASIP Journal on Image and Video Processing*, vol. 2008, p. 246309, May 2008.

[22] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152, ACM, 1992.

[23] L. Leal-Taixé, A. Milan, I. D. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *CoRR*, vol. abs/1504.01942, 2015.

Appendix A

SUMMARY STATISTICS FOR SEQUENCES USED IN OUR EXPERIMENTS

Table 7
Summary Statistics of Training Sequences

Sequences	Frames	Boxes	Cars	Vans	Buses	Other	Weather
20033	784	5,274	4,188	759	327	0	sunny
20034	800	9,934	8,073	488	1,373	0	sunny
20035	800	11,855	10,396	1,459	0	0	sunny
20051	906	8,934	7,738	290	906	0	sunny
20052	694	8,359	6,982	683	694	0	sunny
20061	800	9,256	7,348	1,035	873	0	sunny
20062	800	4,462	3,049	725	501	187	sunny
20063	800	6,392	4,789	712	771	120	sunny
20064	800	14,184	13,478	465	241	0	sunny
20065	1,200	17,156	14,205	1,699	1,252	0	sunny
41063	1,505	10,048	8,447	1,338	76	187	sunny
41073	1,825	10,295	9,143	1,041	0	111	sunny
40871	1,720	36,625	29,634	5,271	1,720	0	rainy
63552	1,150	7,113	5,482	1,540	0	91	rainy
63553	1,405	10,775	9,208	1,338	167	62	rainy
63554	1,445	9,847	8,359	1,457	0	31	rainy
63561	1,285	9,803	8,578	1,007	0	218	rainy
63562	1,185	6,680	5,650	877	66	87	rainy
63563	1,390	9,564	8,322	1,053	31	158	rainy
39931	1,082	3,931	3,495	0	436	0	cloudy
40161	1,490	6,625	4,803	975	847	0	cloudy
40162	1,726	11,261	9,380	454	1,427	0	cloudy
40171	1,150	8,928	7,084	147	1,697	0	cloudy
40172	2,635	18,142	16,671	644	626	201	cloudy
40181	1,700	7,124	4,261	417	2,350	96	cloudy
40191	2,495	38,401	33,647	4,633	0	121	cloudy
40192	2,195	28,183	23,573	4,097	346	167	cloudy
40201	925	10,925	10,141	784	0	0	cloudy
40204	1,225	22,428	19,387	2,774	267	0	cloudy
40211	1,923	6,892	5,792	977	92	31	cloudy
40212	1,690	7,879	6,800	870	188	21	cloudy
40213	1,782	7,702	6,301	1,124	133	144	cloudy
40241	2,320	21,347	18,502	2,596	71	178	cloudy
40243	1,265	10,548	9,047	1,259	171	71	cloudy
40244	1,345	8,811	7,885	804	122	0	cloudy
39761	1,323	3,718	3,316	0	402	0	night
39811	500	599	599	0	0	0	night
39821	880	4,195	3,405	609	181	0	night
39851	1,286	5,126	4,007	342	777	0	night
39861	745	2,394	2,298	0	96	0	night
40732	2,120	11,512	10,154	276	730	352	night
40751	1,145	7,386	5,418	41	1,836	91	night
40752	2,025	16,852	14,529	1,647	479	197	night
40962	1,875	7,580	7,009	475	96	0	night
40963	1,820	11,639	9,906	1,007	726	0	night
40981	1,995	10,349	9,248	990	111	0	night
40991	1,667	4,482	4,482	0	0	0	night
40992	2,122	5,062	4,926	136	0	0	night
Total	**67,745**	**516,577**	**439,135**	**51,315**	**23,205**	**2,922**	-

Appendix B

HYPERPARAMETER CHOICES FOR OUR MODELS

Table 8
Chosen Hyperparameters for the Best PR-MOTA Results (HIOU, IOU, and MDP Trackers)

Tracker	IOU	Max score	Min score	Track length	Historical frames
Overall (Sunny, Night, Cloudy, Rainy)					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.7	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
Sunny					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.7	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Night					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.6	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Cloudy					
HIOU	0.7	0.9	0.0 - 0.9	2	3
IOU	0.7	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
Rainy					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.6	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-

Table 9
Chosen Hyperparameters for the Best PR-MOTP Results (HIOU, IOU, and MDP Trackers)

Tracker	IOU	Max score	Min score	Track length	Historical frames
Overall (Sunny, Night, Cloudy, Rainy)					
HIOU	0.8	0.9	0.0 - 0.9	2	3
IOU	0.8	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Sunny					
HIOU	0.8	0.9	0.0 - 0.9	2	3
IOU	0.8	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Night					
HIOU	0.8	0.9	0.0 - 0.9	2	3
IOU	0.8	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Cloudy					
HIOU	0.8	0.9	0.0 - 0.9	2	3
IOU	0.8	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Rainy					
HIOU	0.8	0.9	0.0 - 0.9	2	3
IOU	0.8	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-

Table 10
Chosen Hyperparameters for the Best PR-MT Results (HIOU, IOU, and MDP Tracker)

Tracker	**IOU**	**Max score**	**Min score**	**Track length**	**Historical frames**
		Overall (Sunny, Night, Cloudy, Rainy)			
HIOU	0.6	0.0	0.0 - 0.9	2	3
IOU	0.3	0.5	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
		Sunny			
HIOU	0.6	0.0	0.0 - 0.9	2	3
IOU	0.3	0.5	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
		Night			
HIOU	0.5	0.0	0.0 - 0.9	2	3
IOU	0.3	0.5	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
		Cloudy			
HIOU	0.6	0.0	0.0 - 0.9	2	3
IOU	0.3	0.5	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
		Rainy			
HIOU	0.6	0.0	0.0 - 0.9	2	3
IOU	0.3	0.5	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-

Table 11
Chosen Hyperparameters for the Best PR-IDS Results (HIOU, IOU, and MDP Tracker)

Tracker	**IOU**	**Max score**	**Min score**	**Track length**	**Historical frames**
		Overall (Sunny, Night, Cloudy, Rainy)			
HIOU	0.5	0.9	0.0 - 0.9	2	3
IOU	0.4	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
		Sunny			
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.5	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
		Night			
HIOU	0.3	0.9	0.0 - 0.9	2	3
IOU	0.4	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
		Cloudy			
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.4	0.9	0.0 - 0.9	2	-
MDP	0.6	-	0.0 - 0.9	-	-
		Rainy			
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.5	0.9	0.0 - 0.9	2	-
MDP	0.6	-	0.0 - 0.9	-	-

Table 12
Chosen Hyperparameters for the Best PR-FM Results (HIOU, IOU, and MDP Tracker)

Tracker	IOU	Max score	Min score	Track length	Historical frames
Overall (Sunny, Night, Cloudy, Rainy)					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.5	0.9	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
Sunny					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.6	0.9	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
Night					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.4	0.9	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
Cloudy					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.5	0.9	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
Rainy					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.6	0.9	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-

Table 13
Chosen Hyperparameters for the Best PR-MOTAL Results (HIOU, IOU, and MDP Tracker)

Tracker	IOU	Max score	Min score	Track length	Historical frames
Overall (Sunny, Night, Cloudy, Rainy)					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.7	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
Sunny					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.8	0.8	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Night					
HIOU	0.6	0.9	0.0 - 0.9	2	3
IOU	0.6	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
Cloudy					
HIOU	0.7	0.9	0.0 - 0.9	2	3
IOU	0.8	0.9	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
Rainy					
HIOU	0.6	0.8	0.0 - 0.9	2	3
IOU	0.7	0.6	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-

Table 14
Chosen Hyperparameters for the Best PR-ML Results (HIOU, IOU, and MDP Tracker)

Tracker	**IOU**	**Max score**	**Min score**	**Track length**	**Historical frames**
			Overall (Sunny, Night, Cloudy, Rainy)		
HIOU	0.6	0.2	0.0 - 0.9	2	3
IOU	0.5	0.6	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
			Sunny		
HIOU	0.6	0.2	0.0 - 0.9	2	3
IOU	0.5	0.6	0.0 - 0.9	2	-
MDP	0.3	-	0.0 - 0.9	-	-
			Night		
HIOU	0.6	0.4	0.0 - 0.9	2	3
IOU	0.5	0.6	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-
			Cloudy		
HIOU	0.7	0.4	0.0 - 0.9	2	3
IOU	0.8	0.7	0.0 - 0.9	2	-
MDP	0.7	-	0.0 - 0.9	-	-
			Rainy		
HIOU	0.7	0.9	0.0 - 0.9	2	3
IOU	0.6	0.9	0.0 - 0.9	2	-
MDP	0.8	-	0.0 - 0.9	-	-

Appendix C

PERFORMANCE SCORES GIVEN DIFFERENT IOU AND MINIMUM CONFIDENCE SCORES

Table 15
Tracking Performance of HIOU (left) and IOU (Right) Trackers on All Test Sequences Given Different Min Confidence Scores, IOU=0.5, and Max Confidence Score=0.6

Score	FAR	IDS	FM	ML
0.0	3.67	533.00	461.00	**2.19**
0.1	1.53	88.00	**208.00**	**2.19**
0.2	1.28	83.00	212.00	**2.19**
0.3	1.12	82.00	229.00	2.73
0.4	0.98	**78.00**	245.00	2.73
0.5	0.88	81.00	286.00	3.28
0.6	0.76	96.00	391.00	3.28
0.7	0.63	96.00	478.00	4.92
0.8	0.49	121.00	497.00	5.46
0.9	**0.33**	134.00	552.00	7.10

Score	FAR	IDS	FM	ML
0.0	11.36	267.00	176.00	**1.64**
0.1	1.65	**82.00**	**150.00**	2.19
0.2	1.25	89.00	157.00	2.19
0.3	1.06	100.00	165.00	2.73
0.4	0.93	126.00	186.00	2.73
0.5	0.85	188.00	236.00	3.28
0.6	0.73	273.00	315.00	4.37
0.7	0.60	327.00	365.00	4.92
0.8	0.47	350.00	378.00	4.92
0.9	**0.32**	390.00	409.00	7.10

Table 16
Tracking Performance of MDP Tracker on All Test Sequences Given Different Confidence Scores (IOU=0.5)

Score	FAR	IDS	FM	ML
0.0	8.47	521.00	310.00	4.37
0.1	1.66	42.00	102.00	**2.73**
0.2	1.38	40.00	112.00	**2.73**
0.3	1.21	42.00	95.00	**2.73**
0.4	1.05	49.00	99.00	3.28
0.5	0.95	40.00	**88.00**	4.37
0.6	0.84	46.00	95.00	4.37
0.7	0.68	55.00	104.00	4.37
0.8	0.58	45.00	107.00	5.46
0.9	**0.41**	**17.00**	109.00	6.56

Table 17
Tracking Performance of HIOU (Left) and IOU (Right) Trackers on All Test Sequences Given Different Min Confidence Scores, IOU=0.6, and Max Confidence Score=0.6

Score	FAR	IDS	FM	ML
0.0	2.20	200.00	**204.00**	2.73
0.1	1.47	166.00	213.00	**2.19**
0.2	1.27	161.00	229.00	**2.19**
0.3	1.13	158.00	251.00	2.73
0.4	1.00	153.00	263.00	2.73
0.5	0.90	**151.00**	301.00	3.28
0.6	0.77	157.00	406.00	3.28
0.7	0.64	**151.00**	489.00	4.37
0.8	0.50	168.00	512.00	4.92
0.9	**0.34**	170.00	562.00	6.56

Score	FAR	IDS	FM	ML
0.0	9.19	311.00	186.00	**2.73**
0.1	1.55	**174.00**	**177.00**	**2.73**
0.2	1.24	176.00	185.00	**2.73**
0.3	1.06	184.00	194.00	**2.73**
0.4	0.93	203.00	213.00	**2.73**
0.5	0.85	256.00	260.00	3.28
0.6	0.74	326.00	331.00	4.37
0.7	0.61	369.00	378.00	4.92
0.8	0.48	386.00	393.00	4.92
0.9	**0.32**	415.00	420.00	7.10

Table 18
Tracking Performance of MDP Tracker on All Test Sequences Given Different Confidence Scores (IOU=0.6)

Score	FAR	IDS	FM	ML
0.0	5.77	52.00	141.00	3.28
0.1	1.62	52.00	95.00	**2.19**
0.2	1.37	61.00	118.00	2.73
0.3	1.18	62.00	112.00	2.73
0.4	1.01	29.00	**88.00**	3.28
0.5	0.93	37.00	92.00	4.37
0.6	0.79	28.00	89.00	3.83
0.7	0.66	34.00	99.00	4.37
0.8	0.54	46.00	106.00	5.46
0.9	**0.39**	**16.00**	117.00	6.56

Table 19
Tracking Performance of HIOU (Left) and IOU (Right) Trackers on All Test Sequences Given Different Min Confidence Scores, IOU=0.7, and Max Confidence Score=0.6

Score	FAR	IDS	FM	ML
0.0	1.75	778.00	**320.00**	**2.73**
0.1	1.37	732.00	345.00	**2.73**
0.2	1.25	722.00	355.00	**2.73**
0.3	1.14	713.00	364.00	**2.73**
0.4	1.03	707.00	386.00	**2.73**
0.5	0.95	696.00	419.00	3.28
0.6	0.82	676.00	512.00	3.83
0.7	0.69	646.00	588.00	4.37
0.8	0.54	625.00	606.00	4.92
0.9	**0.36**	**555.00**	629.00	8.20

Score	FAR	IDS	FM	ML
0.0	6.51	841.00	325.00	**2.73**
0.1	1.46	736.00	**322.00**	**2.73**
0.2	1.23	**733.00**	328.00	**2.73**
0.3	1.09	734.00	332.00	**2.73**
0.4	0.98	743.00	349.00	**2.73**
0.5	0.90	774.00	390.00	3.28
0.6	0.78	815.00	453.00	4.37
0.7	0.66	835.00	493.00	4.92
0.8	0.52	810.00	498.00	4.92
0.9	**0.35**	741.00	494.00	8.74

Table 20
Tracking Performance of MDP Tracker on All Test Sequences Given Different Confidence Scores (IOU=0.7)

Score	FAR	IDS	FM	ML
0.0	5.17	65.00	152.00	3.28
0.1	1.58	32.00	**90.00**	**2.19**
0.2	1.33	34.00	96.00	**2.19**
0.3	1.11	30.00	91.00	2.73
0.4	0.98	36.00	95.00	3.28
0.5	0.89	37.00	93.00	3.83
0.6	0.76	30.00	91.00	4.37
0.7	0.63	51.00	103.00	4.37
0.8	0.54	34.00	115.00	5.46
0.9	**0.38**	**20.00**	127.00	6.56

Table 21
Tracking Performance of HIOU (Left) and IOU (Right) Trackers on All Test Sequences Given Different Min Confidence Scores, IOU=0.8, and Max Confidence Score=0.6

Score	FAR	IDS	FM	ML
0.0	1.52	2,730.00	**849.00**	**5.46**
0.1	1.30	2,651.00	873.00	**5.46**
0.2	1.22	2,640.00	869.00	**5.46**
0.3	1.14	2,627.00	884.00	**5.46**
0.4	1.05	2,612.00	899.00	**5.46**
0.5	0.98	2,589.00	923.00	6.01
0.6	0.85	2,532.00	990.00	7.10
0.7	0.72	2,456.00	1,047.00	7.65
0.8	0.57	2,349.00	1,032.00	9.29
0.9	**0.40**	**2,097.00**	977.00	13.11

Score	FAR	IDS	FM	ML
0.0	3.73	2,598.00	909.00	**5.46**
0.1	1.34	2,530.00	910.00	**5.46**
0.2	1.18	2,522.00	915.00	**5.46**
0.3	1.07	2,521.00	918.00	**5.46**
0.4	0.98	2,522.00	930.00	**5.46**
0.5	0.92	2,529.00	956.00	6.01
0.6	0.80	2,511.00	999.00	7.65
0.7	0.68	2,472.00	1,013.00	8.20
0.8	0.54	2,361.00	983.00	9.29
0.9	**0.38**	**2,122.00**	**902.00**	13.11

Table 22
Tracking Performance of MDP Tracker on All Test Sequences Given Different Confidence Scores (IOU=0.8)

Score	FAR	IDS	FM	ML
0.0	5.65	69.00	167.00	3.28
0.1	1.52	47.00	104.00	**2.19**
0.2	1.24	45.00	104.00	2.73
0.3	1.06	35.00	102.00	3.28
0.4	0.90	31.00	94.00	3.83
0.5	0.84	38.00	98.00	3.83
0.6	0.72	24.00	**86.00**	4.37
0.7	0.62	34.00	102.00	4.92
0.8	0.49	31.00	122.00	5.46
0.9	**0.36**	**20.00**	127.00	6.01