Multi-Object Tracking For Self-Driving Cars

Da Shen, Jing Xie, Lavanya Suresh Kannan, Rui Liu
University of Maryland, College Park

Abstract

Multi-object tracking (MOT) is extremely important to achieve safe autonomous driving. Commonly used way of solving multi-object tracking problem is to use LIDAR, which is accurate but also very expensive. In this project, we established a cost-effective object tracking system that can track multiple objects in a self-driving car dataset using camera information only. Our proposed approach has a detection stage and a tracking stage. In the detection stage, we identify interesting regions that contain objects based on modified Yolo v3 network. In the tracking stage, we identify which object is which, i.e. assigning consistent unique ids to objects over time, by using a motion model and a visual appearance model. The motion model tracks objects via velocity using kalman filtering. The appearance model tracks objects by predicting visual similarity between two cropped object images via feature extraction using Siamese EfficientNet and feature matching using a Relation Network. We find that the object detection network achieves 88.99% mAP score, then appearance model can achieve 98% MOTA score at tracking. Detecting location of objects is the bottleneck for such two-stage approach [1].

1. Introduction

The goal of object tracking is to track each object in every frame of a video by taking an initial set of object detections with a unique ID for each detection, while maintaining the ID all the time. Multi-object tracking is very important to achieve autonomous driving safely, because it is involved in many tasks of autonomous driving, such as path planning, obstacle avoidance and intent recognition [8]. However, multi-object tracking is difficult. A lot of issues can appear due to abrupt object motion, changing appearance patterns of both the object and the scene, nonrigid object structures, object to object and object to scene occlusions, camera motion, and so on [17]. To solve the challenging problem of object tracking, which is a high level spatial reason, some methods have been come up with by previous researchers [7, 16, 19]. One of the most commonly used way is LIDAR, which is accurate but also very expensive at the same time. So in order to solve this problem, We established a cost effective object tracking system that uses camera information only. However, visual 3D detection comes with a major challenge: precision rapidly decreases with increasing distance from the cameras. As a result, the bounding boxes generated by most image-based 3D object detection systems tend to jump around between frames, resulting in an unstable tracking. Some camera-based detections are dropped entirely, leading to tracking loss. So our goal is to predict stable and accurate 3D bounding boxes around multiple vehicles in a self-driving car data set. We aim to provide a vehicle-tracking system that is robust to dropped predictions and more stable frame-to-frame than a pure neural network object detection approach.

2. Related Work

The state-of-the-art in 3D MOT for vehicles is LIDAR [8, 16], which is capable of precise measurements at long range. The drawbacks of LIDAR are mainly cost and complexity, the sensor cost per car is generally very high, and it is computationally expensive to process point cloud videos using neural networks. There have been attempts to achieve the same performance as LIDAR using visual systems. The work of Godard et al. [5] focuses on monocular depth estimation, while the work of Simonelli et al. [10] focuses on monocular 3D localization and orientation for vehicles. However, while orientation prediction has shown good results, the depth-perception and 3D localization precision of such systems has been shown to be extremely lacking compared to LIDAR. Stereo depth-perception systems have shown better precision than monocular depth-perception systems, although it also suffers from imprecision at range [4, 8]. Our work focuses on combining the precision advantage of stereo systems with the principles of monocular tracking and orientation estimation. Wang et al. [15] proposed an MOT system that allows target detection and appearance embedding to be learned in a shared model. They incorporated the appearance embedding model into a single-shot detector, such that the model can simultaneously output detections and the corresponding embeddings. They further proposed a simple and fast association method that can work in conjunction with the joint model.

[1]Our code is available at https://github.com/jingxie0425/CMSC828I_Object_Tracking

3. Approach

Key Idea. Our model has two stages: detection stage and tracking stage. The detection stage is responsible for identifying interesting regions that contain objects. Detection results in the format of bounding boxes serve as the input to the tracking stage. The goal of the tracking stage is to identify which object is which. We have implemented two tracking models:

- **The motion model** tracks the objects by using velocity motion to estimate the future 3D state of the object.
- **The visual appearance model** works by comparing whether two cropped object images visually look similar.

The motion model tracks the objects even in the absence of raw detection while the appearance model relies on the bounding boxes captured by the detection model.

3.1. Details

Object Detection Network. To identify regions of interests and The Yolo v3 [9] detection network is utilized to extract bounding boxes and the categories from the imagery. YOLO v3 is modified to fit the task of object detection in self-driving scenario. The number of categories is reduced from 80 to 10 because we only focus on commonly appeared objects, like different type of vehicles, pedestrians, or bikes, in application scenario on the road instead of a general object detection. We reduced the number of filters in the deep convolutional layers accordingly to 45. The network size is set to default 416 by 416 which works fine.

Instead of learning from scratch, the network is initialized with pre-trained weights on the ImageNet. With the transform learning, we can leverage the advantage of existing models. We also fine-tuned the model to fit the bdd100k datasets. Due to the limited time and computation resources for training, we start the learning rate relative aggressively from 1E-3 to make the model learn fast at the beginning. Then as more data fed to the network, we change the learning rate less aggressively by scaling over steps. So the learning rate starts from 1E-3 and remain constant for 3800 iterations. Then it decreases according to steps policy. To further shorten the training time, we empirically set a lower learning rate at the very beginning by setting the burn-in period 400.

The bdd100k dataset has 2970 images in total. The dataset is split to training (70%), validation (10%), and testing (20%) in two ways. In the first approach, we split the images by scene. There are totally 99 scenes presented in the dataset so we make sure the frames belong to the same scene cannot appeared across the split subsets of data. Thus, there are approximately 70 scenes in training set, 10 scene in validation and 9 scenes in testing set. In the second approach, we put all images together disregard which scene they belong to. And then split them according to the percentage. Besides, to make full usage of this dataset, new data is cooked up by transform the colors of the entire picture using saturation, exposure, and hue.

Motion Model. After the 2D bounding boxes are detected using YOLO, two main steps are followed for tracking the object.

- Calculating the point clouds using stereo: First we are doing camera calibration. Using stereo rectification techniques, the key points are found and matched properly in both the images to calculate re-projection. Using this, we can rectify the images and make sure it is parallel and correctly aligned for our task. Then the disparity map is calculated for the aligned images. Figure 2 shows the disparity map obtained for one image. We are then calculating the depth using the formula below,

  $$Depth = (baseline * focallength)/disparity)$$

  Figure 3 shows the depth map obtained. Using the bounding boxes obtained above, we are extracting the corresponding depth patches for each object. With the help of the depth map we can re project the 2D points in the 3D world.

  $$X = (x - cx) * depth/fx$$
  $$Y = (y - cy) * depth/fy$$
  $$Z = depth$$

  where cx, cy is the center of x, y and fx, fy is the focal length in x - direction and y- direction. For each detected object, we are saving the corresponding gray image patch (uint8) which will be used for matching later on and also save point clouds.

  - Object matching: For each detected vehicle, we have to decide if we are seeing this vehicle for the first time or not. For this, we need some kind of similarity metrics that can be used to compare current vehicle with previously detected vehicles. We have tried several methods including brute force matching approach. we used the results from the appearance model(which is explained in the next section) and calculated the match between two frames. If the current vehicle has no match in the previous frame, we are registering it as a new object. After finding the match between the two frames, we are using Kalman filter as follows to perform the tracking.
  - Kalman filtering: Kalman filter basically has two steps to follow. The first one is prediction and the next one is measurement. For the prediction step we are using the constant velocity model to track the bounding box from previous frame to current frame, and depth values in this bounding box are used to calculate the corresponding 3D point cloud. For the measurement step we are using the Yolo detected output for every object in the current frame. Using the prediction and

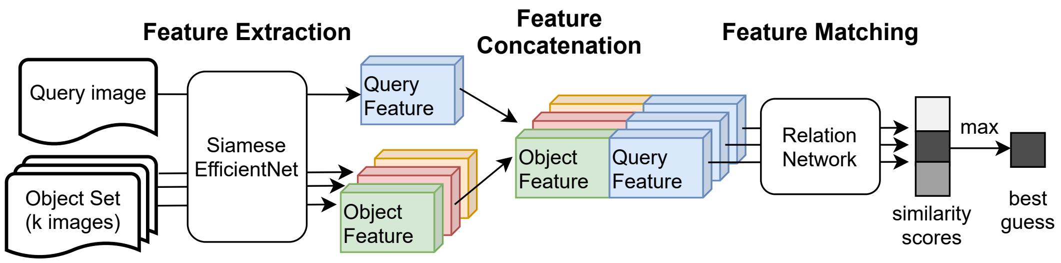

Figure 1. Appearance model architecture. At training time, the training objective is to learn a Relation Network that is able to measure the similarity distance between two input features via a binary classification task over all training samples. At test time, it compares query image with every image in the support set (objects that have occured in the current video), and selects the object that looks most similar to the query image.

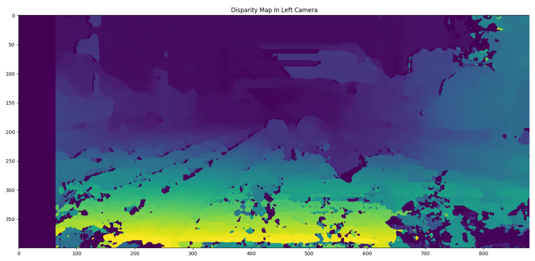

Figure 2. Disparity map

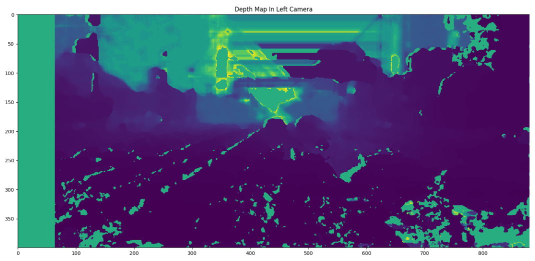

Figure 3. Depth

measurement as input values, we are going to calculate the final estimated location, and update the point clouds accordingly. For vehicles that were present in the previous frame but not present in the current frame, we only do the prediction step and skip the correction. If any vehicle is not detected for at least 3 consecutive frames, it is considered as lost and removed from tracked objects.

Appearance Model. We formulate the object tracking task as a K-way One-shot learning problem:

- Support set (object set): each object that has occurred in the current video is one category. We have one sample image per category.
- Query image: a newly-seen object
- Task: categorizes the query image into k categories of unique objects that have occurred in previous frames.

The appearance model's architecture is shown in Figure 1. It follows the ideas of [11, 13, 14], but we extend it to k-way 1-shot learning where k dynamically varies from video to video and from time to time. This design is because there can be arbitrarily many different objects in a video. We employ Siamese EfficientNet [12] with pre-trained weights for feature extraction. A pair of Siamese neural networks are networks that use same weights while working on two different input to compute comparable output. Then, the feature matching stage implements a relation network [11] that learns to measure the similarity distance between two extracted features. If the new object is not similar to any existing objects in the past frames, then we assign a new id to the new object.

4. Experiments and Results

Dataset. We use a subset of Berkeley Deep Drive (BDD100k) [18] due to limited computational power [2]. The dataset is randomly divided into a training set and a test set. The training set contains 80 videos and the test set contains 20 videos. Each video has 30 frames. The dataset also comes with ground truth object detection and object tracking labels, with a total of 30772 labels.

Evaluation Metric. We use Multiple Object Tracking Accuracy (MOTA) score [2, 6] to measure the quality of our model in the MOT task:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t}$$

where t is frame index, GT_t is the number of objects in ground truth, FN_t is the number of false negative predictions, FP_t is the number of false positive predictions, and $IDSW_t$ is the number of id switches in predictions [2, 6]. An id switch occurs when the same unique object is assigned id i by the model at a frame but another id j at the next frame such that $i \neq j$. For example, if the same car is always assigned a new id for 5 consecutive frames, then the number of id switches is 4. Note that the state-of-the-art result on BDD100k tracking benchmark is only 35.5 mean MOTA (QDTrack) out of 100 [1].

[2]You can download the dataset we used here: https://drive.google.com/file/d/1a7YGLjSTNAK9yA0xLNkTjJya_8zf2vUv/view?usp=sharing.

4.1. Object Detection Network Results

The mean average precision (mAP, same as AP50) [3] is used to evaluate the object detection network. The time used for training Yolo v3 network is 160 epochs (where batch size is 64). Figure 4 demonstrates the mAP improvement on training and testing dataset.

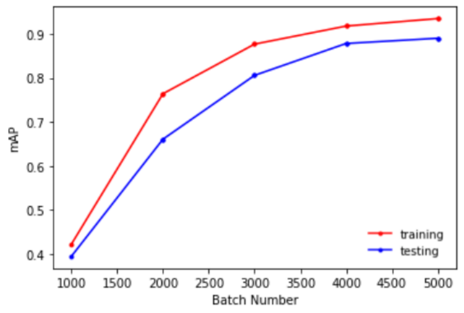

Figure 4. mAP improvement on training and testing datasets during the training process

The baseline model is the trained Yolo v3 model on COCO datasets where there are 80 categories. To compare the baseline model with our trained model, we make two models detect the same dataset. Because the object categories between baseline model and our trained model is not fully overlapped, in terms of the baseline model, we only count the categories that overlap with our model. When the dataset is split using the second approach mentioned in Section 3.1, as shown in Table 1, the trained Yolo v3 model performs much better than the baseline Yolo v3 model on all categories. The mAP reaches 88.99% on the test dataset, much higher than the baseline.

Object	car	truck	bus	trailer	other vehicle	motorcycle	bicycle	pedestrian	rider	otherperson	mAP
Yolo v3 Trained AP	91.90%	93.11%	94.02%	100.00%	87.48%	100.00%	80.85%	79.92%	62.63%	100.00%	88.99%
Yolo v3 Baseline	57.78%	18.58%	13.63%			13.81%	22.31%	35.01%			26.85%

Table 1. The mAP 50 of trained Yolo v3 model compared with the baseline Yolo v3 model

When data is split by the first approach mentioned in Section 3.1, however, the Object detection network performs poorly on test dataset with only 12% mAP, though still outperforms the baseline model in car and truck categories which appear most frequently in our data. The training mAP is as good as the result presented above. The problem looks like an overfitting issue but it actually doesn't help when penalizing more to large value of weights. The reason might be the small size of the dataset which makes the training not robust across different scene. The training just make the model "memorize" some of the familiar scene shown in the training set but has no idea of a purely new scene. In the second data split strategy, where the data is randomly shuffled despite the scene, training model is most likely to see different frames of the scene that appeared in the test dataset, making the trained model more robust.

4.2. Motion Model Results

If tracking is not done and if we rely only on Yolo detection, we lose tracked objects whenever Yolo fails to recognize them. Then if these objects are detected in new frames again, they are registered as new objects. With Kalman filter tracking, we keep tracking objects even if Yolo misses them in some frames. As shown in Figure 5., Blue bounding box represents the final estimated location of vehicle, and red bounding box represents the vehicle that Yolo missed and yet successfully tracked by KF.

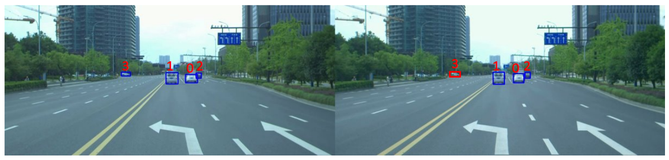

Figure 5. Stereo image pair. In right image, Yolo missed the object detection, but tracker was able to predict its location.

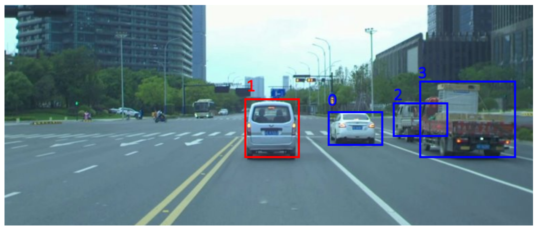

Figure 6. Yolo missed the object detection in current frame, but tracker was able to predict it's current location from previous frame.

4.3. Appearance Model Results

Experiment Setup. The experiment contains a training phase and a test phase. During the **training phase**, we learn a Relation Network that is able to measure the similarity distance between two input features. We train the Relation Network through a binary classification task where a training sample is a pair of cropped object images in two consecutive frames labeled with whether the two objects are the same or not. The training samples and labels are generated using all training videos and ground truth object id labels in the dataset. We experimented with two relation network architecture:

- RelationNN_fc: input feature $\rightarrow$ average pooling $\rightarrow$ fully connected layer $\rightarrow$ fully connected layer
- RelationNN_conv: input feature $\rightarrow$ 3×3 conv layer $\rightarrow$ average pooling $\rightarrow$ 2×2 conv layer $\rightarrow$ fully connected layer $\rightarrow$ fully connected layer

with batch norm, ReLU activation, and dropout after each layer except the output layer. For the output layer, we apply sigmoid activation.

During the **test phase**, we evaluate the appearance model on two tasks: 2-way 1-shot evaluation and k-way 1-shot evaluation (the real tracking task). In the 2-way 1-shot task, the model is given a pair of cropped object images and it is asked to determine whether they are the same object or not. It mimics the training phase and is not the real object tracking task. We calculate an accuracy score by the number of correct predictions divided by the total number of all samples.

In the k-way evaluation task, the setting is the real-world object tracking task where we are given frames of videos and bounding boxes of objects detected in each frame by the object detection module, and the task is to assign consistent unique object ids to each unique object. We calculate the MOTA score over all the objects in each video.

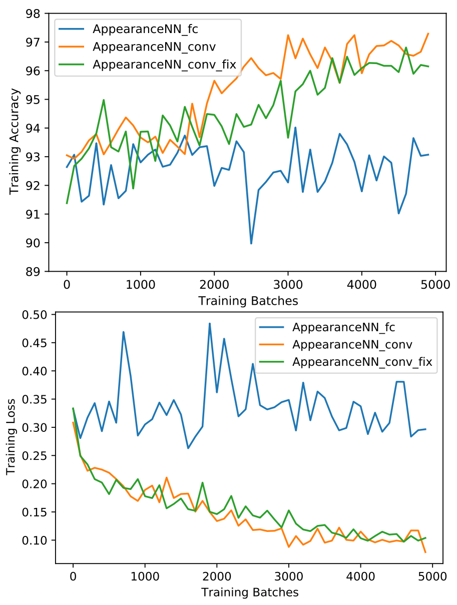

Figure 7. Training accuracy and loss of the appearance model over batches. The plotted accuracy/loss is an average over 100 consecutive batches. Batch size is 16. One epoch contains 20997 batches. AppearanceNN_fc denotes that the relation network contains fully connected layers only. AppearanceNN_conv denotes that the relation network contains convolution layers. AppearanceNN_conv_fix denotes that the feature network is fixed and we only train the relation network.

Experiment Results and Takeaway. The results of first 5000 batches of the training phase is presented in Figure 7. We can observe that using fully connected layers does not work well because loss is not decreasing over batches. A Relation Network with convolution layers works better than one using fully connected layers only. Training the feature extraction network at the same time yields a lightly higher accuracy than training the relation network only while fixing the feature extraction network.

Task	Test Score	Metric
2-way	98.2% after 1 epoch of training 98.1% after 2 epoches of training	accuracy
k-way	97.0% after 1 epoch of training	MOTA

Table 2. Test scores of the appearance model. The k-way 1-shot task is the real tracking task. For detailed task setup, please refer to the experiment setup paragraph.

For the test phase, results are presented in Table 2. We want to focus on the model's ability of tracking objects instead of proposing regions that contain objects, so we use ground truth location (bounding boxes) of objects as input. In the 2-way task, we observed that training for a second epoch does not help improve test accuracy. In the k-way 1-shot (the real tracking task) task, the model achieves 97.0% MOTA score. It means that the appearance models performs very well at tracking which object is which (i.e. comparing visual appearance of objects between frames and assigning consistent unique ids to the objects throughout the video), if we can accurately detect location (bounding boxes) of objects. Detecting objects location is the bottleneck.

5. Contribution

- Jing: The object detection model. Provide inputs for the tracking models.
- Lavanya: The object tracking model using linear Kalman Filter. Used appearance matching to to match and track vehicles detected by Yolo.
- Da: The appearance model. Preparing the dataset and a short notebook to get the team started.
- Rui: Literature review. Introduction. Related Work.

6. Conclusion

In this paper, we proposed a two-stage object tracking model using object detector and tracking models. One advantage of our approach is that the object detection module, the motion model, and the appearance model work independently. It means that we can improve each of the modules independently without worrying about the rest of the pipeline. One disadvantage of the approach is that the object detector can not use additional information about object trajectory obtained by tracking models to assist in detection. Thoroughly blending the two stages together may help the overall performance.

Problems Encountered.

- Limited computational power. No GPU cluster access. Self-driving car datasets are usually very large (>10GB).

- Finding an appropriate dataset for the model was a big task initially.
- Difficulties with real-world images: Some objects are very small. Lighting is changing. Motion blur is severe sometimes.
- The Object detection network is not robust across different scenes
- Intrinsic matrix provided by the open source stero dataset was not accurate enough, it needs stereo rectification. We did not get enough time for implementing stereo rectification pipeline, as a result depth maps were not accurate enough.
- Tuning semi global matching parameters for disparity calculation took a lot of time.
- Clustering all object point clouds and displaying in single scene was not feasible given the time.

Future Work.

- 2 weeks. We plan to use better training tricks and finetune object detection module and appearance model.
- 2 months. For the motion model, we will track full 3D bounding box for each object. For the appearance model, we will extend from 1-shot setting to few-shot setting by considering all previous appearance of the same object. For the detection model, make it more robust by training on larger datasets.
- 1 semester. We will extend the appearance model to take entire camera image as input and extracts Regions of Interest (RoI Pooling) to find similar objects. In this way, tracking models do not need to rely on detection module.

References

[1] Multiple object tracking leaderboard on bdd100k, 2021. `https://paperswithcode.com/sota/multiple-object-tracking-on-bdd100k`. 3

[2] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP J. Image Video Process.*, 2008, 2008. 3

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 4

[4] R. Ginhoux and J.-S. Gutmann. Model-based object tracking using stereo vision. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1226–1232 vol.2, 2001. 1

[5] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, 2017. 1

[6] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016. 3

[7] Addie Ira Borja Parico and Tofael Ahamed. Real time pear fruit detection and counting using yolov4 models and deep sort. *Sensors*, 21(14), 2021. 1

[8] Akshay Rangesh and Mohan Manubhai Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars. *IEEE Transactions on Intelligent Vehicles*, 4(4):588–599, 2019. 1

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[10] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. *CoRR*, abs/1905.12365, 2019. 1

[11] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[12] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. 3

[13] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 3

[14] Guangting Wang, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. Spm-tracker: Series-parallel matching for real-time visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[15] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *CoRR*, abs/1909.12605, 2019. 1

[16] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, 2013. 1

[17] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. 38(4), 2006. 1

[18] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[19] Lin Zhao, Meiling Wang, Sheng Su, Tong Liu, and Yi Yang. Dynamic object tracking for self-driving cars using monocular camera and lidar. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10865–10872, 2020. 1