Joint Monocular 3D Vehicle Detection and Tracking

Hou-Ning Hu[1], Qi-Zhi Cai[2], Dequan Wang[3], Ji Lin[4],
Min Sun[1], Philipp Krähenbühl[5], Trevor Darrell[3], Fisher Yu[3]
[1]National Tsing Hua University [2]Nanjing University [3]UC Berkeley [4]MIT [5]UT Austin

Abstract

3D vehicle detection and tracking from a monocular camera requires detecting and associating vehicles, and estimating their locations and extents together. It is challenging because vehicles are in constant motion and it is practically impossible to recover the 3D positions from a single image. In this paper, we propose a novel framework that jointly detects and tracks 3D vehicle bounding boxes. Our approach leverages 3D pose estimation to learn 2D patch association overtime and uses temporal information from tracking to obtain stable 3D estimation. Our method also leverages 3D box depth ordering and motion to link together the tracks of occluded objects. We train our system on realistic 3D virtual environments, collecting a new diverse, large-scale and densely annotated dataset with accurate 3D trajectory annotations. Our experiments demonstrate that our method benefits from inferring 3D for both data association and tracking robustness, leveraging our dynamic 3D tracking dataset.

1. Introduction

Autonomous driving motivates much of contemporary visual deep learning research. Driven by progress in monocular object detection and scene segmentation, it promises to make low-cost mobility widely available. However, many commercially successful approaches to autonomous driving control rely on an array of views and sensors, reconstructing 3D point clouds before inferring object trajectories in 3D. In contrast, human observers have no difficulty to perceive the 3D world in both space and time from simple sequences of 2D images rather than 3D point cloud, even though human stereo vision only reaches several meters. In this paper, we explore architectures and datasets sufficient to develop similar capabilities using deep neural networks.

Monocular 3D detection and tracking is also inherently ill posed. In the absence of a depth measurements or strong priors, a single view does not provide enough information to accurately estimate 3D layout of a scene. Without a good layout estimate, tracking is increasingly hard, especially in the presence of large ego-motion (*e.g.* a turning car). The two problems are inherently intertwined. Good tracking helps 3D detection, as information along consecutive frames is integrated. Good 3D detection helps tracking, as ego-motion can be factored out.

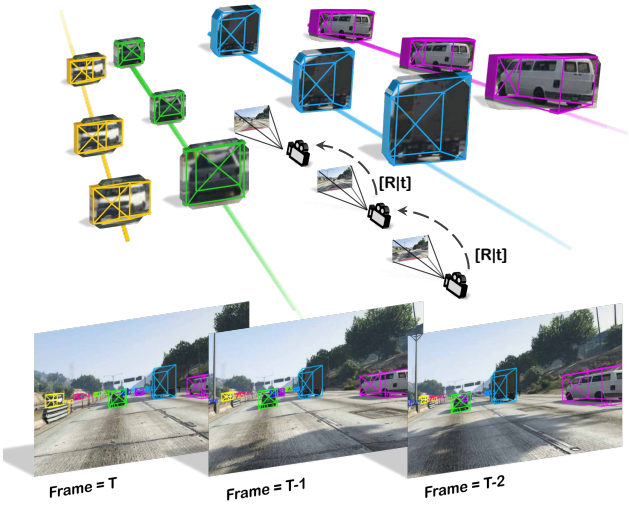


Figure 1: Joint detection and tracking in 3D. Predicting 3D bounding box trajectories of the observed vehicles in image sequences captured by a monocular camera requires a large amount of data. Our dataset provides rich annotations for this new task. Our algorithm pipeline predicts the corresponding 3D bounding boxes based on potential regions for vehicles and jointly forms 3D trajectories by tracking those regions across frames.

In this paper, we propose a deep network architecture to track and detect vehicles jointly in 3D from a series of monocular color images. Figure 1 provides the overview of our 3D tracking and detection task. After detecting 2D bounding box of targets, we utilize both world coordinates and re-projected camera coordinates to associate instances cross frames. Notably, we leverage novel occlusion-aware association and depth-ordering matching algorithms to overcome occlusion and reappearance problem in tracking. Fi-

nally we capture the movement of instances in a world coordinate system and update 3D pose based on each single-frame estimation along a trajectory, integrating observations associated with the instance over time.

Like any deep network, our model is data hungry. The more data we feed it, the better it performs. However, existing datasets are either limited to static scenes [45], lack the required ground truth trajectories [29], or are too small to train contemporary deep models [12]. To bridge this gap, we resort to realistic video games. We use a new pipeline to collect large-scale 3D trajectories, from a realistic synthetic driving environment, augmented with dynamic meta-data associated with each observed scene and object. The dataset is visually diverse and rich enough to capture a wide range of scene appearances and dynamics with accurate 3D tracking annotations.

To the best of our knowledge, we are the first to tackle the estimation of complete 3D vehicle bounding box tracking information from monocular camera. We jointly track the vehicles across frames based on 2D appearance and predicted 3D positions and estimate the full 3D information of the tracks including position, orientation, and dimensions. Our experiments show that 3D position can improve predicted location in new frame than traditional 2D tracking, and estimating 3D positions for the whole track is more accurate than single-frame estimation. The depth orders of the tracked vehicles construct an important cue to reduce mismatch rate.

2. Related Works

Object tracking has been explored extensively in the last decade [48, 40, 43]. Early methods [5, 11, 23] track objects based on correlation filters. Recent ConvNet-based methods typically build on pre-trained object recognition networks. Some generic object trackers are trained entirely online, starting from the first frame of a given video [16, 1, 21]. However, these online training methods cannot fully utilize the large amount of video data. Held *et al*. [18] proposed a regression based method for offline training of neural networks, tracking novel objects at test-time at 100 fps. Siamese networks also found use in tracking, including tracking by object verification [44], tracking by correlation [3], tracking by detection [10]. Yu *et al*. [47] enhance tracking by modeling a track-let into different state and explicitly learns a MDP for state transition. Due to the absence of 3D information, it just use 2D location to decide whether a track-let is occluded.

All those methods only take 2D visual features into consideration, where the search space is restricted near the original position of the object. This works well for a static observer, but fails in a dynamic 3D environment. Here, we further leverage 3D information to narrow down the search space, and stabilize the trajectory of target objects.

Sharma *et al*. [42] uses 3D cues for 2D vehicle tracking. Scheidegger *et al*. [41] also adds 3D kalman filter on the 3D positions to get more consistent 3D localization results. Because the goals are for 2D tracking, 3D box dimensions and orientation are not considered. Osep *et al*. [31] and Li *et al*. [26] studies 3D bounding box tracking with stereo cameras. Because the 3D depth can be perceived directly, the task is much easier, but in many cases such as ADAS, large-baseline stereo vision is not possible.

Object detection reaped many of the benefits from the success of convolutional representation. There are two mainstream deep detection frameworks: 1) two-step detectors: R-CNN [14], Fast R-CNN [13], and Faster R-CNN [35]. 2) one-step detectors: YOLO [32], SSD [27], and YOLO9000 [33]. We apply Faster R-CNN, one of the most popular object detectors, as our object detection input. The above all algorithms all rely on scores of labeled images to train on. In 3D tracking, this is no different. The more training data we have, the better our 3D tracker performs. Unfortunately, getting a large amount of 3D tracking supervision is hard.

Driving datasets have attracted a lot of attention in recent years. KITTI [12], Cityscapes [7], Oxford RobotCar [28], and BDD100K [51] provide well annotated ground truth for visual odometry, stereo reconstruction, optical flow, scene flow, object detection and tracking. However, their provided 3D annotation is very limited. Accurate 3D annotations are challenging to obtain from humans and expensive to measure with 3D sensors like LiDAR. Therefore these real-world public datasets are typically small in scale or poorly annotated.

To overcome this difficulty, there has been significant work on virtual driving datasets: virtual KITTI [11], SYNTHIA [38], GTA5 [37], VIPER [36], CARLA [8], and Free Supervision from Video Games (FSV) [22]. The previous works have proven realistic simulation is very effective for collecting large-scale photo-realistic images with high-quality annotations and pushing advance of research study. We extend FSV [22] to include object tracking in both 2D and 3D, as well as fine-grained object attributes, control signals from driver actions. We use a synthetic autopilot to play the game while we collect data. The closest dataset to ours is VIPER [36], which provides a suite of videos and annotations for various computer vision problems. Our dataset focuses on object tracking and provide 40 times of tracks for large-scale training and evaluation.

In the next section, we describe how to generate 3D object trajectories from 2D dash-cam videos. Considering the practical requirement of autonomous driving, we primarily focus on online tracking systems, where only the past and current frames are accessible to a tracker.

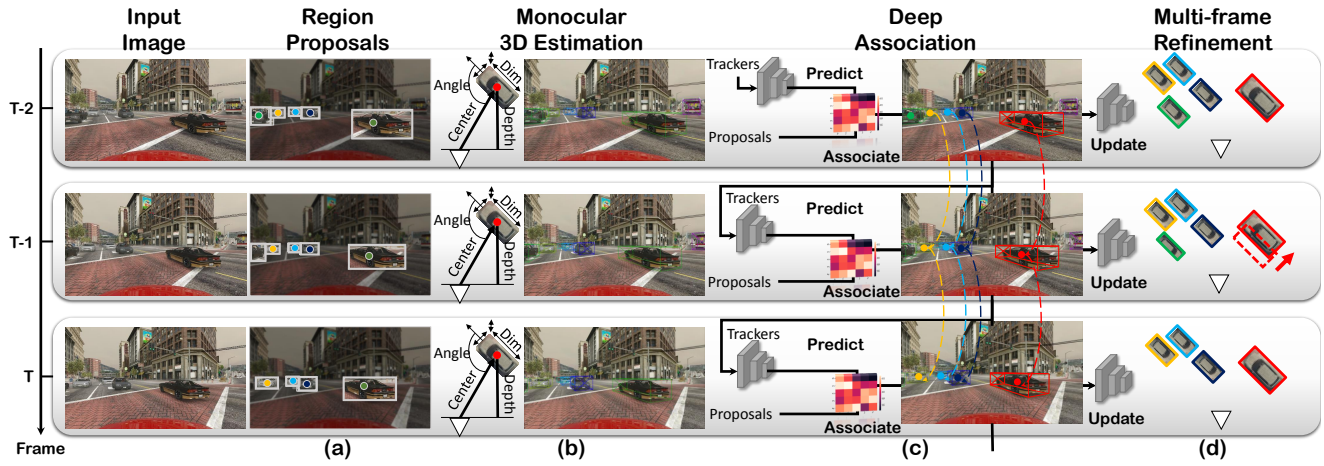


Figure 2: Overview of our learning framework. Our proposed model processes an input image to candidate RoIs (a). For each region of interest, we jointly learn 2D bounding box detection and 3D layout estimation (b). Our 3D estimation provides association a more robust prediction (c). With the help of 3D tracking, we further improves the ability of 3D estimation by associating object features of the previous frames (d).

3. Joint 3D Detection and Tracking

Our goal is to track objects and infer their precise 3D location, orientation, and dimension from a single monocular video stream. For each image, we use an off-the-shelf detector to detect all candidate objects. We model 3D information with a layer-aggregating network on the object proposals. We leverage estimated 3D information of current trajectories to track them through time, using 3D re-projection to generate similarity metric between all trajectories and detected boxes. Our method should also solve the occlusion problem with the help of occlusion-aware data association and depth-ordering matching. Finally, we re-estimate the 3D location of the object using the newly matched trajectory.

3.1. Problem Formulation

We phrase the 3D tracking problem as a supervised learning problem. We aim to find M trajectories $\{\mathbf{t}_1, \ldots, \mathbf{t}_M\}$, one for each objects in a video. Each trajectory $\mathbf{t}_i$ is a sequence of states $\{t_a^{(i)}, t_{a+1}^{(i)}, \ldots, t_b^{(i)}\}$ starting at the first visible frame a and ending at the last visible frame b. The state of an object at frame a is given by

$$t_a^{(i)} = (P, O, D, F, \Delta P),$$

where P defines the 3D world location (x, y, z) of the object, and ΔP stands for its velocity $(\dot{x}, \dot{y}, \dot{z})$. O, D, F denotes for object orientation θ_l, dimension (l, w, h) and appearance feature f, respectively.

The whole system is powered by a convolutional network pipeline trained on very large amount of ground truth supervision. Figure 2 shows an overview of our system. Next, we discuss each component in more detail.

3.2. Candidate Box Detection

We employ Faster R-CNN [35] trained on our dataset to provide bounding boxes of object proposals. The detection results are used to locate the candidate vehicles and extract their appearance features. However, the 3D box center is usually not projected to the center of 2D boxes. So we have to estimate the 3D box center specifically for better accuracy.

3D box center projection. To estimate 3D layout from single image more accurately, we extends the design of Region Proposal Network (RPN) [35] to hypothesize a projected 2D point from 3D bounding box center, which reduce the gap between 2D bounding box center and projected 3D box center.

Specifically, raw images are fed into a deep layer-aggregated ConvNet to generate global convolutional feature maps. With extended RPN head, the model regresses both a bounding box and a 3D center location from a anchor point. Then, we use ROIalign [17] instead of ROIpool to get the regional representation given the detected regions of interest (ROIs). This reduces the misalignment of two-step quantization. Each ROI (Figure 2(a)) corresponds to a 2D bounding box $\{x_{\texttt{min}}, y_{\texttt{min}}, x_{\texttt{max}}, y_{\texttt{max}}\}$, a 3D box center projection (x_c, y_c), a confidence score c and a corresponding feature vector f, which can be used in 3D estimation.

3.3. 3D Box Estimation

We estimate complete 3D box information (Figure 2(b)) based on an ROI position on image coordinate and a feature representation of the bounding box in question. The ROI feature vector f is extracted from a 34-layer DLA-up [50] with ROIalign. To estimate 3D layout from single image, we

attach a sub-network, which extends the design of Mousavian *et al*. [30], on the top of ROI features. The network is trained ground truth depth, dimension and orientation estimates each by a 3-layers 3x3 convolution module to preserve spatial information.

Contrast to previous approaches, we further infer depth from monocular images. We regress an inverse depth value $1/d$ but minimize the depth value d and its projected 3D location P using L1 loss. A 3D location P is projected from an estimated 3D object center on an image (x_c, y_c) using predicted depth and camera transformation.

Given the coordinate distance $\hat{x} = x_c - \frac{w}{2}$ to the horizontal center of an image and the focal length f, we can easily restore the global rotation θ in the camera coordinate from θ_l with simple geometry, $\theta = (\theta_l - \arctan \frac{\hat{x}}{f}) \mod 2\pi$. Following MultiBin [30] for θ_l estimation, we first classify the alpha into two bins and then regress the residual relative to the bin center using Smooth L1 loss.

In the driving scenario, the high variance in dimension distribution of different category vehicles (e.g., car, bus) results in a difficulty of classification using unimodal object proposals. Therefore, we regress a dimension D to the ground truth dimension over the object feature representation using L1 loss.

The 3D estimation provides us with an observation for location P with object orientation θ_l and dimension D.

However we do not have direct access to a measurement of the distance z a tracklet is at. While the depth could, in theory, be inferred directly from the box coordinate and the ego-motion information, we found this inaccurate in a dynamic scenario.

For any new tracklet, we train the network to predict monocular depth of the object, by leveraging the location and ROI features. For any tracked object, the network is able to learn a mixture of a monocular and stereoscopic 3D estimate by merging the location from last visible frames and a current frame. The next step is to generate such 3D trajectory of each tracked object in world coordinates.

3.4. Data Association and Tracking

Given a set of tracks $\{\mathbf{t}_K, \ldots, \mathbf{t}_N\}$ at frame a, our goal is to associate each track with a candidate detection, spawn new tracks, or end a track (Figure 2(c)).

We solve the data association problem by a weighted bipartite matching algorithm, on affinities based on an overlap between current trajectory projected forward in time and candidates of bounding boxes, and a learned similarity of deep appearance representation. First, we project each trajectory forward in time, using the estimated velocity of an object and the camera ego-motion. Here, we assume that ego-motion is given by a sensor, like GPS, an accelerometer, gyro and/or IMU. The re-projection provides us with a 2D bounding box $\mathbf{t}_i = \{x_{\texttt{min}}, y_{\texttt{min}}, x_{\texttt{max}}, y_{\texttt{max}}\}$ for each trajectory.

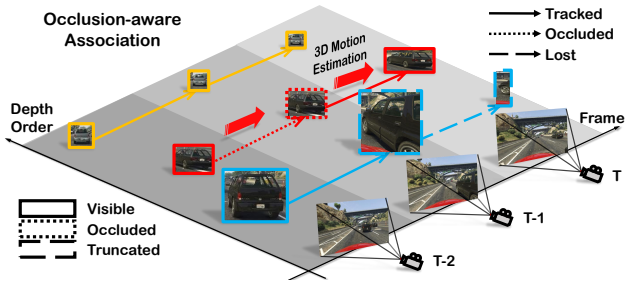


Figure 3: Illustration of Occlusion-aware association. A tracked tracklet (yellow) is visible all the time, while a tracklet (red) is occluded by another (blue) at frame $T-1$. During occlusion, the tracklet doesn't update state but keep inference motion until reappearance. For a truncated or disappear tracklet (blue at frame T), we left it as lost.

We define an affinity matrix $\mathbf{A}(\mathbf{t}_i, \mathbf{d}_j)$ between a track $\mathbf{t}_i$ and candidate $\mathbf{d}_j$ as a joint probability of appearance and location correlation between their corresponding objects.

$$\mathbf{A}_{\texttt{app}}(\mathbf{t}_i, \mathbf{d}_j) = \exp(-||f_{\mathbf{t}_i}, f_{\mathbf{d}_j}||_1) \tag{1}$$

$$\mathbf{A}_{\texttt{2D}}(\mathbf{t}_i, \mathbf{d}_j) = \frac{\mathbf{t}_i \cap \mathbf{d}_j}{\mathbf{t}_i \cup \mathbf{d}_j} \tag{2}$$

$$\mathbf{A}_{\texttt{3D}}(\mathbf{t}_i, \mathbf{d}_j) = \frac{\mathbf{M}(X_{\mathbf{t}_i}) \cap \mathbf{M}(X_{\mathbf{d}_j})}{\mathbf{M}(X_{\mathbf{t}_i}) \cup \mathbf{M}(X_{\mathbf{d}_j})}, \tag{3}$$

where $X_{\mathbf{t}_i}$ and $X_{\mathbf{d}_j}$ are the tracked and predicted 3D bounding box, and $\mathbf{M}$ is the projection matrix casting the bounding box to image coordinates.

$$\begin{aligned}\mathbf{A}(\mathbf{t}_i, \mathbf{d}_j) =& w_{\texttt{app}}\mathbf{A}_{\texttt{app}}(\mathbf{t}_i, \mathbf{d}_j) + w_{\texttt{2D}}\mathbf{A}_{\texttt{2D}}(\mathbf{t}_i, \mathbf{d}_j) \\ &+ w_{\texttt{3D}}\mathbf{A}_{\texttt{3D}}(\mathbf{t}_i, \mathbf{d}_j),\end{aligned} \tag{4}$$

where $w_{\texttt{app}}, w_{\texttt{2D}}, w_{\texttt{3D}}$ are the weights of appearance, 2D overlap, and 3D overlap. We utilize a mixture of those factors as the affinity across frames, similar to the design of POI [49].

Comparing to 2D tracking, 3D-oriented tracking is more robust to visual occlusion, overlapping, and re-appearances. When a target is temporally occluded, the corresponding 3D motion estimator can roll-out for a period of time and relocate 2D location at each time via camera coordinate transformation.

Occlusion-aware Data Association. Similar to previous state-of-the-art methods [46, 47, 39], we model the lifespan of a tracker into 4 major subspaces in MDP state space: birth, tracked, lost and death. Each unmatched detection spawns a new tracklet, however, an unmatched tracklet is not immediately terminated. Tracklets naturally disappear in occluded region and reappear later. We keep unmatched tracklets around for 10 time-steps before they die out.

We address the dynamic object inter-occlusion problem by separating a new state called "Occluded" from a Lost

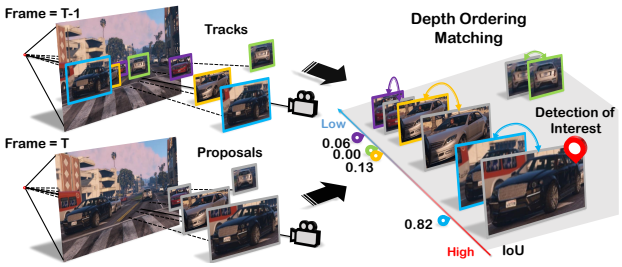


Figure 4: Illustration of Depth-ordering matching. Given the tracklets and detections, we sort them into a list by depth order. For each detection of interest (DOI), we calculate the IOU between DOI and non-occluded tracklets regions. The depth ordering naturally provides higher probabilities to tracklets near the DOI.

state. An object is occluded by another object in the front with over 70% region would fall into Occluded state. An occluded tracklet will not update its lifespan nor its feature representation until the reappearance is clear from occlusion. During the occluded state, the life time-step of a tracklet holds unchanged, yet we still predict its 3D location using the estimated motion. Figure 3 illustrates how the occlusion state works.

Depth-Ordering Matching. We introduce instance depth ordering in assigning a detection to neighbor tracklets, which modeling a strong prior of relative depth ordering in human perception. For each detection, we insert it into depth ordered tracklets one by one. By calculating Intersection of Union (IoU) of non-occluded region of tracklets from the view of a detection of interest, it naturally provides higher probabilities of linking neighbor tracklets than ones a few layers away. Figure 4 shows depth ordering pipeline. In this way, we obtain data association of moving objects with the help of 3D trajectories in world coordinates. Finally, we solve data association using the Kuhn-Munkres algorithm. In the next subsection, we show how to estimate that distance leveraging the associated tracklet and bounding box using a deep network.

3.5. Motion Model

Deep Motion Estimation and Update. To exploit the temporal consistency of certain vehicles, we associate the information across frames by using two LSTMs. Given the current location of observation from 3D estimation module, the updating LSTM considers both observed and previous predicted location and updates the location and velocity (Figure 2(c)). The predicting LSTM models dynamic object location in 3D coordinates by estimating object velocity from previous updated location X_{T-1} and current possible location $\hat{X}_T$. This allows for handling missed and occluded objects by finding motion in 3D world coordinates, which naturally cancels out adverse effects of ego-motion.

The LSTMs keep updating the state from the observation of matched detection, while simply predicting via linear velocity model if there is no matched bounding box. Therefore, we model 3D motion (Figure 2(d)) in world coordinates allowing occluded tracklet to move along motion plausible paths while managing the birth and death of moving objects.

Concretely, our pipeline consists of a single-frame monocular 3D object detection model for object-level pose inference and recurrent neural networks for inter-frame object association and matching. We extend the region processing to include 3D estimation by employing multi-head modules for each object instance. In data association, we introduced occlusion-aware association to solve inter-object occlusion problem. For tracklet matching, depth ordering lowers mismatch rate by filtering out distant candidates from a target. The motion estimator updates the velocity of each object, assuming a linear velocity model, independent of camera movement or interactions with other objects. The final pipeline produces accurate and dense object trajectories in 3D world coordinate system.

4. 3D Vehicle Tracking Simulation Dataset

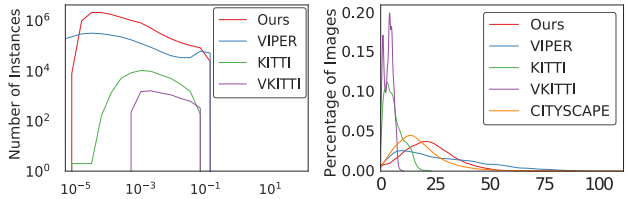
It is laborious and expensive to annotate large-scale 3D bounding box image dataset even in the presence of LiDAR data, although it is much easier to label 2D bounding boxes on tens of thousands of videos [51]. Therefore, no such dataset collected from real sensors is available to the research community. To resolve the data problem, we turn to driving simulation to obtain accurate 3D bounding box and 2D instance segmentation annotations at no cost of human efforts. Our data collection and annotation pipeline extend the previous works like VIPER [36] and FSV [22], especially in terms of linking identities across frames. Compared to the others, our dataset has more diversity regarding instance scales (Figure 5a) and closer instance distribution to real scenes (Figure 5b).

Our simulation is based on *Grand Theft Auto V*, a modern game that simulates a functioning city and its surroundings in a photo-realistic three-dimensional world. To associate object instances across frames, we utilize in-game API to capture global instance id and corresponding 3D annotations directly. In contrast, VIPER leverages a weighted matching algorithm based on a heuristic distance function, which can lead to inconsistencies. It should be noted that our pipeline is real-time, which provides the potential of large-scale data collection, while VIPER requires expensive off-line processing. Furthermore, more data can be collected online for the specific driving policy that is currently being learned. This is especially important in combination with reinforcement or imitation learning.

Table 1 demonstrates the detailed comparison with related

Dataset	Task	Object	Frame	Track	Boxes	3D	Weather	Occlusion	Ego-Motion
KITTI-D [12]	D	C,P	7k	-	41k	✓		✓	-
KAIST [20]	D	P	50k	-	42k		✓	✓	-
KITTI-T [12]	T	C	8k	938	65k	✓		✓	✓
MOT15-3D [24]	T	P	974	29	5632	✓	✓		
MOT15-2D [24]	T	P	6k	500	40k		✓		
MOT16 [29]	T	C,P	5k	467	110k		✓	✓	
UA-DETRAC [45]	D,T	C	84k	6k	578k		✓	✓	
Virtual KITTI [11]	D,T	C	21k	2610	181k	✓	✓	✓	✓
VIPER [36]	D,T	C,P	184k	8272	5203k	✓	✓	✓	✓
Ours	D,T	C,P	688k	325k	10068k	✓	✓	✓	✓

Table 1: Comparison to related dataset for detection and tracking (Upper half: real-world, Lower half: synthetic). We only count the size and types of annotations for training and validation (D=detection, T=tracking, C=car, P=pedestrian). To our knowledge, our dataset is the largest 3D tracking benchmark for dynamic scene understanding, with control signals of driving, sub-categories of object.



(a) Instance scale (b) Instances per image

(c) Frames of tracking (d) Vehicle distance

Figure 5: Statistical summary of our dataset in comparison of KITTI [12], VKITTI [11], VIPER [36], and Cityscapes[7]

datasets, including detection, tracking, and driving benchmarks. KITTI-D [12] and KAIST [20] are mainly detection datasets, while KITTI-T [12], MOT15 [24], MOT16 [29], and UA-DETRAC [45] are primarily 2D tracking benchmarks. The common drawback could be the limited scale, which cannot meet the growing demand for training data. Compared to related synthetic dataset, Virtual KITTI [11] and VIPER [36], we additionally provide fine-grained attributes of object instances, such as color, model, maker attributes of vehicle, motion and control signals, which leaves the space for imitation learning system. More details on a comparison to dataset statistics can be found in the appendix.

5. Experiments

We train and evaluate our 3D detection and tracking pipeline on KITTI MOT benchmark and our large-scale dataset, featuring real-world driving scenes and a wide variety of road conditions in a diverse virtual environment, respectively.

3D Estimation. We adapt depth evaluation metrics [9] from image-level to object-level, leveraging both error metrics, such as absolute relative difference (Abs Rel), squared relative difference (Sq Rel), root mean square error (RMSE) and RMSE log, and accuracy metrics, % of y_i that $max(y_i/y_i^*, y_i^*/y_i) < \delta$ where $\delta = 1.25, 1.25^2, 1.25^3$. For orientation evaluation, we follow the setting of KITTI [12], using orientation score (OS) for each object. We propose two metrics for evaluating estimated object dimension and 3D projected center position. A Dimension Score (DS) measures how close an object volume estimation to a ground truth. DS is defined as

$$DS = \min(\frac{V_{\texttt{pred}}}{V_{\texttt{gt}}}, \frac{V_{\texttt{gt}}}{V_{\texttt{pred}}})$$

with an upper bound 1, where V is the volume of a 3D box by multiplying its dimension $l * w * h$. A Center Score (CS) measures distance of a projected 3D center and a ground truth. CS is calculated by

$$CS = (1 + \cos(a_{\texttt{gt}} - a_{\texttt{pd}}))/2,$$

with a upper bound 1, where a depicts an angular distance $((x_{\texttt{gt}} - x_{\texttt{pd}})/w_{\texttt{pd}}, (y_{\texttt{gt}} - y_{\texttt{pd}})/h_{\texttt{pd}})$, weighted by corresponding box width and height in the image coordinates.

Object Tracking. We follow the set of evaluation metrics from CLEAR [2], including false negative (FN), track precision (TP), track recall (TR), miss-match rate (MM), multiple object tracking accuracy (MOTA) and multiple Object Tracking Precision (MOTP).

Overall Evaluation. We evaluated the performance of 3D layout estimation by calculating 3D IoU mAP with refined depth estimation of different tracking methods. The metric reflects the conjunction of all 3D components, dimension, rotation and depth, and tracking accuracy.

Motion	Deep	Order	MOTA↑	MOTP↑	TP↑	TR↑	MM↓	FN↓
-	-	-	5.056	67.726	79.445	71.102	6.310	62.637
2D [4]	-	-	57.042	82.794	86.905	77.058	3.739	33.134
2D* [46]	✓	-	65.892	81.860	90.607	87.186	4.796	19.213
3D	-	-	69.616	84.776	85.947	84.202	1.435	21.298
3D	✓	-	69.843	84.523	90.242	87.113	2.269	18.547
3D	-	✓	70.061	**84.845**	85.952	84.377	**1.317**	21.319
3D	✓	✓	70.126	84.494	90.155	**87.432**	2.123	18.177
LSTM	✓	✓	**70.439**	84.488	**90.651**	87.374	1.959	**18.094**
-	-	-	4.609	71.064	**74.941**	83.594	14.535	49.531
2D [4]	-	-	42.748	81.878	70.003	84.292	9.172	31.350
2D* [46]	✓	-	48.518	81.479	66.381	88.222	7.270	22.989
3D	-	-	54.324	83.914	64.986	90.869	3.032	19.070
3D	✓	-	54.855	83.812	68.235	91.687	2.087	17.835
3D	-	✓	54.738	**84.045**	64.001	90.623	3.242	19.899
3D	✓	✓	**55.218**	83.751	68.628	**92.132**	**1.902**	**17.302**
LSTM	✓	✓	55.150	83.780	69.860	92.040	2.150	17.470

Table 2: Comparison of tracking performance with different methods and features in GTA dataset. The upper half of the table show evaluation of targets within 100 meters while the lower within 150 meters. 3D motion with the help of our depth estimation is more robust in linking candidate tracklets than 2D motion. Using Deep feature in correlation can reduce the false negative (FN) rate. Using depth-order matching scheme filters out possible mismatching trajectories. Replacing 3D kalman filter with LSTM module to re-estimate the location using temporal information. Although 3D motion reach higher MOTA in 150m, LSTM benefits 3D IoU AP in Table 3.

Method	Easy			Medium			Hard		
	$AP_{3d}70$	$AP_{3d}50$	$AP_{3d}25$	$AP_{3d}70$	$AP_{3d}50$	$AP_{3d}25$	$AP_{3d}70$	$AP_{3d}50$	$AP_{3d}25$
None	6.13	35.12	69.52	4.93	24.25	53.26	4.05	17.26	41.33
3D	6.14	35.15	69.56	4.94	24.27	53.29	4.06	17.27	41.42
LSTM	**7.89**	**36.37**	**73.39**	**5.25**	**26.18**	**53.61**	**4.46**	**17.62**	**41.96**

Table 3: Comparison of tracking performance on 3D IoU AP 25, 50, 70. Noted that 3D Kalman filter slightly improves the AP performance compare to single-frame estimation (None), while Linear (LSTM) grants a clear margin. The difference comes from how we model object motion in the 3D coordinate. Instead of using Kalman filter smoothing between prediction and observation, we directly model vehicle movement using LSTM.

5.1. Implementation

Training. We train our DLA [50] network using the full training set with detection bounding boxes from Faster R-CNN. DLA produces the feature map as input of ROIalign [17]. The training is conducted for 100 epochs using Adam optimizer with initial learning rate 10^{-3}, momentum 0.9, and weight decay 10^{-4}. Each GPU has 5 images and each image with no more than 300 candidate objects before NMS. We train on 4 GPUs (effective mini-batch size is 20) while keep the original image resolution, 1920×1080. The anchors of RPN in Faster R-CNN span 4 scales and 3 ratios, in order to detect the small objects at a distance.

Dataset. Our raw data is recorded at 12 FPS, which is helpful for temporal aggregation. With the goal of autonomous driving in mind, we focus on vehicles closer than $150m$, and also filtered out the bounding boxes whose areas are smaller than 256 pixels. The dataset is then split into train, validation and test set with ratio $10:1:4$.

5.2. Results

3D for Tracking. We compare with Sort [4], Deep Sort [46] tracker for inter-frame ROIs matching in 2D. The detailed numbers of tracking performance could be found in Table 2. We observe the performance improvement, especially for track recall (TR) and Accuracy (MOTA), from 2D to 3D with predicted depth, which indicates 3D information helps to recover unmatched pairs of objects. Moreover, adding deep appearance feature distinguishes two near-overlapping objects, which reduces mismatch rate (MM) in the most cases. With occlusion-aware association and depth-ordering matching, our false negative (FN) rate drops with an observable margin.

Tracking for 3D. We conduct experiments on 3D AP to investigate how tracking benefits 3D layout estimation. By comparing single frame estimation with 3D Kalman filter and motion model (LSTM), we observe that tracking benefits 3D estimation with temporal information. The number tells that 3D Kalman filter provides a small improvement via trajectory smoothing within prediction and observation. On the other hand, our LSTM modules give a learned estimation based on input prediction and observation, which provides more room in improvements.

Real-world Evaluation. Besides evaluating on synthetic data, we resort to KITTI tracking benchmark to compare our model abilities. Results are listed on Table 5. To make a fair comparison, we use RRC [34] detection results following a published state-of-the-art method, BeyondPixels [42]. Since the 2D detection does not come with 3D projected center, we revert to 2D bounding box center. Although it could degenerate our performance, our tracking pipeline outperforms current published online methods.

Amount of Data Matters. We train depth estimation module with 1%, 10%, 30% and 100% training data. The results show how we can benefit from more data in Table 4, where there is a consistent trend of performance improvement as the amount of data increases. The trend of our results with different amount of training data indicates that large-scale 3D annotation is helpful, especially with accurate ground truth of far and small objects.

The results on matching between frames show that we can obtain a steady growth in accuracy with a robust 3D estimation. In return, the results on 3D IoU AP show that

Dataset	Amount	Depth Error Metric				Depth Accuracy Metric			Orientation	Dimension	Center
		Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSElog ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	OS ↑	DS ↑	CS ↑
GTA	1%	0.258	4.861	10.168	0.232	0.735	0.893	0.934	0.811	0.807	0.982
	10%	0.153	3.404	6.283	0.142	0.880	0.941	0.955	0.907	0.869	0.982
	30%	0.134	3.783	5.165	0.117	0.908	0.948	0.957	0.932	0.891	0.982
	100%	**0.112**	**3.361**	**4.484**	**0.102**	**0.923**	**0.953**	**0.960**	**0.953**	**0.905**	**0.982**
KITTI	100%	0.074	0.449	2.847	0.126	0.954	0.980	0.987	0.962	0.918	0.974

Table 4: Performance of 3D box estimation. The depth, orientation, dimension and center metrics demonstrate the effectiveness of our model from each separate metrics. The models are trained and tested on the same dataset, either GTA or KITTI. With different amount of training data in our GTA dataset, the results suggest that large data capacity benefits the performance of a data-hungry network.

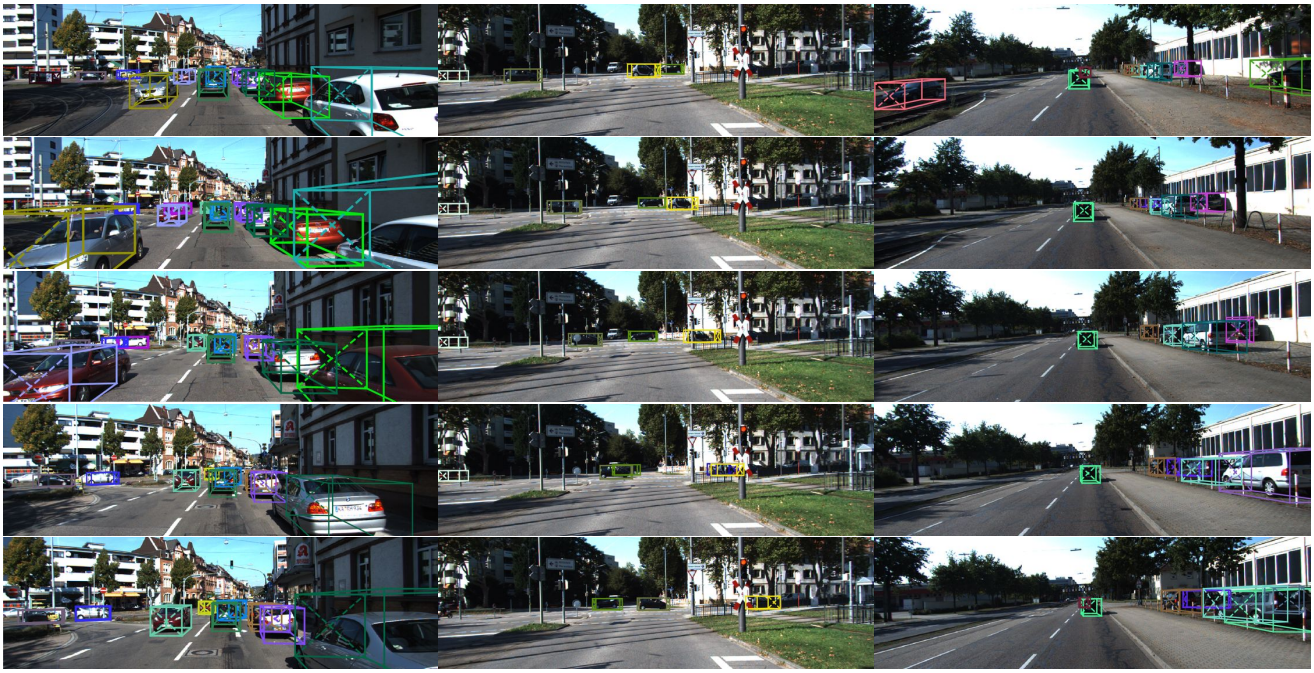


Figure 6: Qualitative results of 3D Estimation on KITTI testing set. We show predicted 3D layout colored with tracking index.

Benchmark	MOTA ↑	MOTP↑	MT ↑	ML↓	FP↓	FN↓
Ours	**84.52**	85.64	**73.38**	**2.77**	705	**4242**
BeyondPixels [42]	84.24	**85.73**	73.23	**2.77**	705	4247
PMBM [41]	80.39	81.26	62.77	6.15	1007	5616
extraCK [15]	79.99	82.46	62.15	5.54	642	5896
MDP [47]	76.59	82.10	52.15	13.38	**606**	7315
SCEA [19]	75.58	79.39	53.08	11.54	1306	6989
CIWT [31]	75.39	79.25	49.85	10.31	954	7345
NOMT-HM* [6]	75.20	80.02	50.00	13.54	1143	7280
mbodSSP [25]	72.69	78.75	48.77	8.77	1918	7360

Table 5: Comparison on the testing set of KITTI tracking benchmark. Only published online methods are reported.

tracking benefits 3D estimation. Moreover, the results on real-world scenario demonstrate that the effectiveness of our pipeline.

6. Conclusion

In this paper, we learn 3D vehicle dynamics from monocular videos. We propose a novel framework, combining spatial visual feature learning and global 3D state estimation, to track moving vehicles in 3D world. We obtain a large-scale densely annotated imagery dataset of synthetic driving scenes, which provides accurate 3D trajectory annotations fully automatically. Both qualitative and quantitative results indicate that our model takes advantage of 3D estimation leveraging our collection of dynamic 3D trajectories.

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, 2009. 2

[2] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *JIVP*, 2008. 6

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016. 2

[4] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *ICIP*, 2016. 7

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2

[6] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE international conference on computer vision*, pages 3029–3037, 2015. 8

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 6

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017. 2

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 6

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *ICCV*, 2017. 2

[11] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 2, 6

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 6

[13] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[15] G. Gündüz and T. Acarman. A lightweight online multiple object vehicle tracking method. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 427–432. IEEE, 2018. 8

[16] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. *TPAMI*, 2016. 2

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 3, 7

[18] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 2

[19] J. Hong Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon. Online multi-object tracking via structural constraint event aggregation. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 1392–1400, 2016. 8

[20] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. *Integrated Comput.-Aided Eng*, 2013. 6

[21] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 2012. 2

[22] P. Krähenbühl. Free supervision from video games. In *CVPR*, 2018. 2, 5

[23] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder. The visual object tracking vot2015 challenge results. In *ICCVW*, 2015. 2

[24] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015. 6

[25] P. Lenz, A. Geiger, and R. Urtasun. Followme: Efficient online min-cost flow tracking with bounded memory and computation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4364–4372, 2015. 8

[26] P. Li, T. Qin, and a. Shen. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[28] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 year, 1000 km: The oxford robotcar dataset. *IJRR*, 2017. 2

[29] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 2, 6

[30] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 4

[31] A. Osep, W. Mehner, M. Mathias, and B. Leibe. Combined image- and world-space tracking in traffic scenes. In *ICRA*, 2017. 2, 8

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[33] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017. 2

[34] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu. Accurate single stage detector using recurrent rolling convolution. In *CVPR*, 2017. 7

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 3

[36] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *ICCV*, 2017. 2, 5, 6

[37] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 2

[38] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2

[39] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 300–311, Oct 2017. 4

[40] S. Salti, A. Cavallaro, and L. Di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *TIP*, 2012. 2

[41] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstrom. Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018. 2, 8

[42] S. Sharma, J. A. Ansari, J. Krishna Murthy, and K. Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 2, 7, 8

[43] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *TPAMI*, 2014. 2

[44] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 2

[45] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *arXiv preprint arXiv:1511.04136*, 2015. 2, 6

[46] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 4, 7

[47] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 4705–4713, 2015. 2, 4, 8

[48] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 2006. 2

[49] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. Poi: multiple object tracking with high performance detection and appearance feature. In *ECCV*, 2016. 4

[50] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3, 7

[51] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018. 2, 5

A. 3D Center Projection

We estimate the vehicle 3D location by first estimating the depth of the detected 2D bounding boxes. Then the 3D position is located based on the observer's pose and camera calibration. We find that accurately estimating the vehicle 3D center and its project is critical for accurate 3D box localization. However, the 2D bounding box center can be far away from the 3D center projection for several reasons. First, there is always a shift from the 2D box center if the 3D bounding box is not axis aligned in the observer's local coordinate system. Second, the 2D bounding box is detecting the visible area of the objects after occlusion and truncation, while the 3D bounding box is defined based on the object's physical dimensions. The projected 3D center can be even out of the detected 2D boxes. These situations are illustrated in Figure 7.

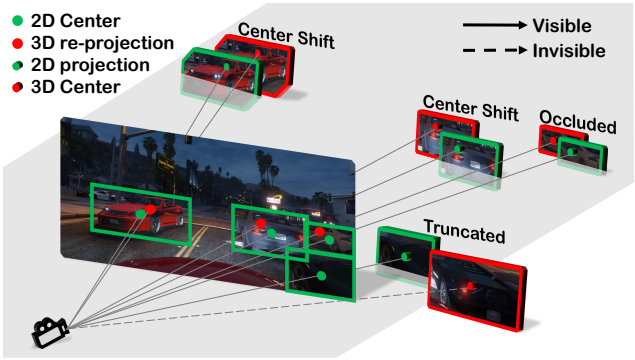


Figure 7: The illustration of the difference in 2D, 3D projected center. A visible 2D center projection point may wrongly locate above or under the ground plane in the 3D coordinates. Other states, occluded or truncated, would inevitably suffer from the center shift problem. The center shift causes misalignment for GT and predicted tracks and harms 3D IOU AP performance.

B. Data Association Details

Occlusion-Aware Association. During association, we set apart occluded tracklets from associating until their estimated locations have a match to current detection. Occluded state helps reduce identity switch when objects are crossing each other. Although holding the life time-step from dying out, we sweep out an occluded object to Death state when estimated 3D location reaches over the boundary of prediction range $-10m$ to $150m$. The sweeping scheme benefits tracker from consuming a huge amount of memory to store all the occluded tracks.

Depth-Ordering Matching. To calculate 3D affinity scores, we sort trackers and detections into layers by depth order. The Intersection of Union (IOU) of the non-occluded region of tracklets is calculated from each side of the detection of interest (DOI). Closer tracklets blocks farther ones in calculating the area of overlapping. So each DOI matches to the tracklet with highest IOU while implicitly models a strong prior of depth sensation called "Occultation".

C. Motion Model

In our experiments, we compare a variant of approaches in estimating locations. 2D Kalman filter models

$$\{x, y, s, a, \Delta x, \Delta y, \Delta a\}$$

in 2D coordinates, where s stands for the width/high ratio and a for the area of the object. It often misses tracking due to substantial dynamic scene changes from sudden ego-motion. We also propose a 3D Kalman filter which models

$$\{x, y, z, \Delta x, \Delta y, \Delta z\}$$

in 3D coordinates. However, it updates in between observation and prediction location and tries to minimize a square mean of the error, while LSTM doesn't be bounded in such a restriction. The proposed final motion model consists of two LSTMs, one for predicting and the other for updating. The latter is in charge of refining 3D locations. Predicting LSTM learns a velocity $\{\Delta x, \Delta y, \Delta z\}$ using hidden state based on the previous observation and predicts a smoothed step toward an estimated location. Updating LSTM takes both observed and previous predicted location as input, and updates the 3D location and velocity from hidden state leveraging previous observations.

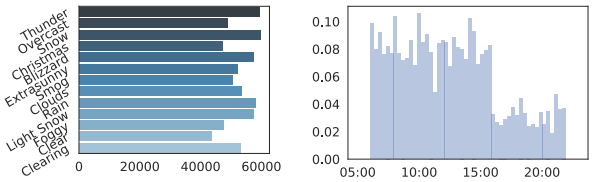
D. Dataset Statistics

To help understand our dataset and its difference, we show more statistics.

Weather and Time of Day. Figure 8 shows the distribution of weather, hours of our dataset. It features a full weather cycle and time of a day in a diverse virtual environment. By collecting various weather cycles (Figure 8a), our model learns to track with a higher understanding of environments. With different times of a day (Figure 8b), the network handles changeable perceptual variation.

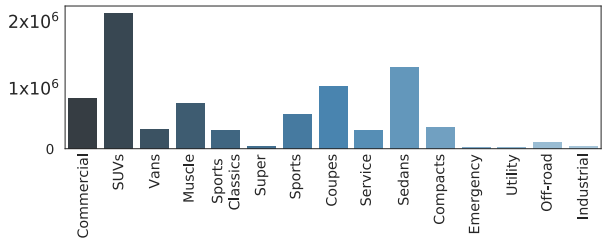
Number of Instances in Each Category. The vehicle diversity is also very large in the GTA world, featuring 15 fine-grained subcategories. We analyzed the distribution of the 15 subcategories in Figure 9a. Besides instance categories, we also show the distribution of occluded (Figure 9c) and truncated (Figure 9b) instances to support the problem of partially visible in the 3D coordinates.

Examples of Our Dataset. Figure 10 shows some visual examples in different time, weather and location.

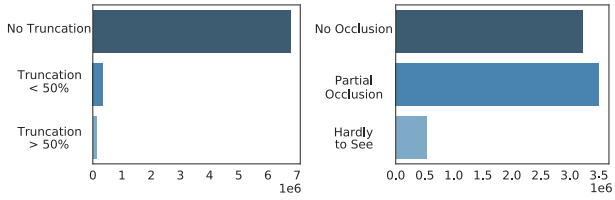


(a) Weather (b) Hours

Figure 8: The statistics of scene in our dataset.



(a) Category



(b) Truncation (c) Occlusion

Figure 9: The statistics of object in our dataset.

E. Experiments

Data Association Weights. During our experiment, we use different weights of appearance, 2D overlap, and 3D overlap for corresponding methods. We select weight ratios based on the results of our validation set.

For None and 2D, we use $w_{\texttt{app}} = 0.0, w_{\texttt{2D}} = 1.0, w_{\texttt{3D}} = 0.0$. For 3D related methods, we give $w_{\texttt{3D}} = 1.0$ and $w_{\texttt{2D}} = 0.0$. For Deep related methods, we times $w_{\texttt{2D}}$ or $w_{\texttt{3D}}$ with 0.7 and 0.3 for $w_{\texttt{app}}$. For Occlusion related methods, we set weights as $w_{\texttt{app}} = 0.3, w_{\texttt{2D}} = 0.0, w_{\texttt{3D}} = 1.0$. We set $w_{\texttt{3D}}$ to 1.0 since occlusion of objects will lower overlapping ratio drastically.

Qualitative results. We show our evaluation results in Figure 11 on our GTA test set.

Evaluation Video. We've uploaded a demo video which summaries our main concepts and demonstrates video inputs with estimated bounding boxes and tracked trajectories in bird's eye view. We show the comparison of our method with baselines in both our GTA dataset and real-world videos.

Figure 10: Examples of our dataset.

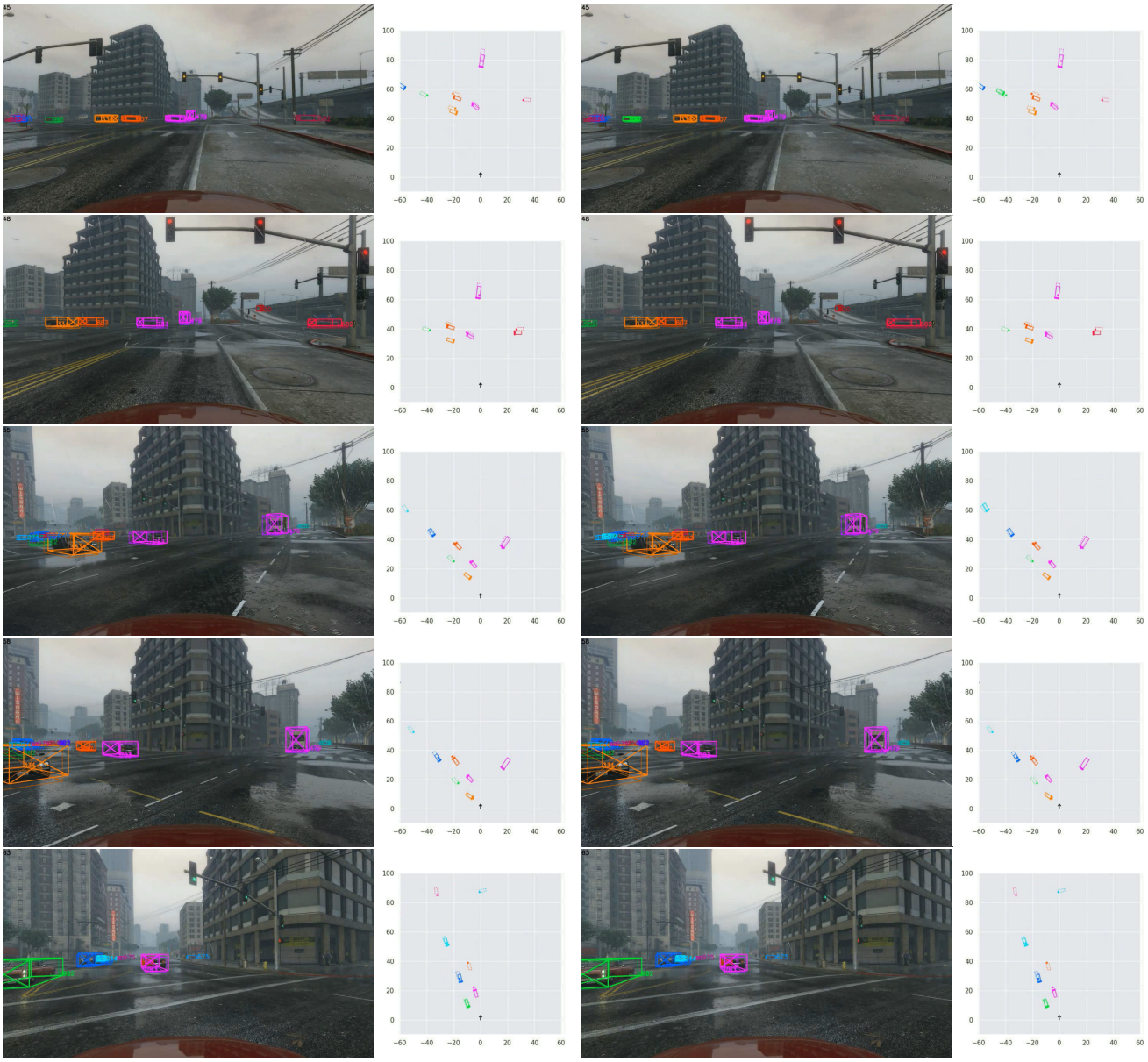


Figure 11: Qualitative comparison results of 3D Kalman and our method.