Joint Monocular 3D Vehicle Detection and Tracking

Hou-Ning Hu1, Qi-Zhi Cai2, Dequan Wang3, Ji Lin4, Min Sun1, Philipp Krähenbühl5, Trevor Darrell3, Fisher Yu3
1National Tsing Hua University 2Sinovation Ventures AI Institute 3UC Berkeley 4MIT 5UT Austin

Abstract

Vehicle 3D extents and trajectories are critical cues for predicting the future location of vehicles and planning future agent ego-motion based on those predictions. In this paper, we propose a novel online framework for 3D vehicle detection and tracking from monocular videos. The framework can not only associate detections of vehicles in motion over time, but also estimate their complete 3D bounding box information from a sequence of 2D images captured on a moving platform. Our method leverages 3D box depth-ordering matching for robust instance association and utilizes 3D trajectory prediction for re-identification of occluded vehicles. We also design a motion learning module based on an LSTM for more accurate long-term motion extrapolation. Our experiments on simulation, KITTI, and Argoverse datasets show that our 3D tracking pipeline offers robust data association and tracking. On Argoverse, our image-based method is significantly better for tracking 3D vehicles within 30 meters than the LiDAR-centric baseline methods.

1. Introduction

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

* Work was done while Hou-Ning Hu, Qi-Zhi Cai and Ji Lin were at the Berkeley DeepDrive Center
ation and depth-ordering matching algorithms to overcome the occlusion and reappearance problems in tracking. Finally, we capture the movement of instances in a world coordinate system and update their 3D poses using LSTM motion estimation along a trajectory, integrating single-frame 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 [41], lack the required ground truth trajectories [26], or are too small to train contemporary deep models [13]. 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.

To the best of our knowledge, we are the first to tackle the estimation of complete 3D vehicle bounding box tracking information from a monocular camera. We jointly track the vehicles across frames based on deep features and estimate the full 3D information of the tracks including position, orientation, dimensions, and projected 3D box centers of each object. The depth ordering of the tracked vehicles constructs an important perceptual cue to reduce the mismatch rate. Our occlusion-aware data association provides a strong prior for occluded objects to alleviate the identity switch problem. Our experiments show that 3D information improves predicted association in new frames compared to traditional 2D tracking, and that estimating 3D positions with a sequence of frames is more accurate than single-frame estimation.

2. Related Works

Object tracking has been explored extensively in the last decade [44, 36, 39]. Early methods [4, 12, 21] 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, 19]. A typical tracker will sample patches near the target object which are considered as foreground and some farther patches as background. These patches are then used to train a foreground-background classifier. However, these online training methods cannot fully utilize a 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 in use, including tracking by object verification [40], tracking by correlation [3], tracking by detection [11]. Yu et al. [43] enhance tracking by modeling a track-let into different states and explicitly learns an Markov Decision Process (MDP) for state transition. Due to the absence of 3D information, it just uses 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. [38] uses 3D cues for 2D vehicle tracking. Scheidegger et al. [37] 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. [28] and Li et al. [22] 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 [15], Fast R-CNN [14], and Faster R-CNN [31]. 2) one-step detectors: YOLO [29], SSD [24], and YOLO9000 [30].

We apply Faster R-CNN, one of the most popular object detectors, as our object detection input. The above 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 [13], Cityscapes [8], Oxford RobotCar [25], BDD100K [47], NuScenes [5], and Argoverse [6] 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 compared to virtual datasets. Accurate 3D annotations are challenging to obtain from humans and expensive to measure with 3D sensors like LiDAR. Therefore these real-world datasets are typically small in scale or poorly annotated.

To overcome this difficulty, there has been significant work on virtual driving datasets: virtual KITTI [12], SYNTHIA [34], GTA5 [33], VIPER [32], CARLA [9], and Free Supervision from Video Games (FSV) [20]. The closest dataset to ours is VIPER [32], which provides a suite of videos and annotations for various computer vision problems while we focus on object tracking. We extend FSV [20] to include object tracking in both 2D and 3D, as well as fine-grained object attributes, control signals from driver actions.

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.

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 and a GPS sensor. Figure 2 shows an overview
of our system. Images are first passed through a detector network trained to generate object proposals and centers. These proposals are then fed into a layer-aggregating network which infers 3D information. Using 3D re-projection to generate similarity metric between all trajectories and detected proposals, we leverage estimated 3D information of current trajectories to track them through time. Our method also solves the occlusion problem in tracking with the help of occlusion-aware data association and depth-ordering matching. Finally, we re-estimate the 3D location of objects using the LSTM through the newly matched trajectory.

3.1. Problem Formulation

We phrase the 3D tracking problem as a supervised learning problem. We aim to find $N$ trajectories $\{\tau^1, \ldots, \tau^N\}$, one for each object in a video. Each trajectory $\tau^i$ links a sequence of detected object states $\{s_a^{(1)}, s_{a+1}^{(1)}, \ldots, s_b^{(1)}\}$ 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 $s_a = (P, O, D, F, \Delta P)$, where $P$ defines the 3D world location $(x, y, z)$ of the object, and $\Delta P$ stands for its velocity $(\dot{x}, \dot{y}, \dot{z})$. $O, D, F$ denotes for object orientation $\theta$, dimension $(l, w, h)$ and appearance feature $f_{app}$, respectively. In addition, we reconstruct a 3D bounding box $X$ for each object, with estimated $P, O, D$ and the projection $c = (x_c, y_c)$ of 3D box’s center in the image. The bounding boxes enable the use of our depth-ordering matching and occlusion-aware association. Each bounding box $X$ also forms a projected 2D box $M(X) = \{x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}\}$ projected onto a 2D image plane using camera parameters $M = K[R[t]]$.

The intrinsic parameter $K$ can be obtained from camera calibration. The extrinsic parameter $[R[t]]$ can be calculated from the commonly equipped GPS or IMU sensor. The whole system is powered by a convolutional network pipeline trained on a considerable amount of ground truth supervision. Next, we discuss each component in more detail.

3.2. Candidate Box Detection

In the paper, we employ Faster R-CNN [31] trained on our dataset to provide object proposals in the form of bounding boxes. Each object proposal (Figure 2(a)) corresponds to a 2D bounding box $d = \{x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}\}$ as well as an estimated projection of the 3D box’s center $c$. The detection results are used to locate the candidate vehicles and extract their appearance features. However, the centers of objects’ 3D bounding boxes usually do not project directly to the center of their 2D bounding boxes. As a result, we have to provide an estimation of the 3D box center for better accuracy. More details about the estimation of the 3D center can be found in the supplementary material\(^1\).

**Projection of 3D box center.** To estimate the 3D layout from single images more accurately, we extend the regression process to predict a projected 2D point of the 3D bounding box’s center from an ROI-pooled feature $F$ using L1 loss. Estimating a projection of 3D center is crucial since a small gap in the image coordinate will cause a gigantic shift in 3D. It is worth noting that our pipeline can be used with any off-the-shelf detector and our 3D box estimation module is extendable to estimate projected 2D points even if the detector is replaced. With the extended ROI head, the model

---

\(^1\)Supplementary material of Joint Monocular 3D Vehicle Detection and Tracking can be found at https://eborbohuc.github.io/Mono-3DT/
regress both a bounding box $d$ and the projection of 3D box's center $c$ from an anchor point. ROIalign [17] is used instead of ROIPool to get the regional representation given the detected regions of interest (ROIs). This reduces the misalignment of two-step quantization.

### 3.3. 3D Box Estimation

We estimate complete 3D box information (Figure 2(b)) from an ROI in the image via a feature representation of the pixels in the 2D bounding box. The ROI feature vector $F$ is extracted from a 34-layer DLA-up [46] using ROIalign. Each of the 3D information is estimated by passing the ROI features through a 3-layer 3x3 convolution sub-network, which extends the stacked Linear layers design of Mousavian et al. [27]. We focus on 3D location estimation consisting of object center, orientation, dimension and depth, whereas [27] focus on object orientation and dimension from 2D boxes. Besides, our approach integrates with 2D detection and has the potential to jointly training, while [27] crops the input image with pre-computed boxes. This network is trained using ground truth depth, 3D bounding box center projection, dimension, and orientation values. A convolutional network is used to preserve spatial information. A convolutional network is used to replace another architecture, the center $c$ can be obtained from this sub-network.

#### 3D World Location

Contrasting with previous approaches, we also infer 3D location $P$ from monocular images. The network regresses an inverse depth value $1/d$, but is trained to minimize the L1 loss of the depth value $d$ and the projected 3D location $P$. A projected 3D location $P$ is calculated using an estimated 2D projection of the 3D object center $c$ as well as the depth $d$ and camera transformation $M$.

**Vehicle Orientation.** 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 restore the global rotation $\theta$ in the camera coordinate from $\theta_l$ with simple geometry, $\theta = (\theta_l + \arctan(\frac{\hat{x}}{f}))$ mod $2\pi$. Following [27] for $\theta_l$ estimation, we first classify the angle into two bins and then regress the residual relative to the bin center using Smooth L1 loss.

**Vehicle Dimension.** In driving scenarios, the high variance of the distribution of the dimensions of different categories of vehicles (e.g., car, bus) results in difficulty classifying various vehicles using unimodal object proposals. Therefore, we regress a dimension $D$ to the ground truth dimension over the object feature representation using L1 loss.

The estimation of an object’s 3D properties provides us with an observation for its location $P$ with orientation $\theta$, dimension $D$ and 2D projection of its 3D center $c$. For any new tracklet, the network is trained to predict monocular object state $s$ of the object by leveraging ROI features. For any previously tracked object, the following association network is able to learn a mixture of a multi-view monocular 3D estimates by merging the object state from last visible frames and the current frame. First, we need to generate such a 3D trajectory for each tracked object in world coordinates.

### 3.4. Data Association and Tracking

Given a set of tracks $\{\tau_1, \ldots, \tau^K\}$ at frame $a$ where $1 \leq J \leq K \leq M$ from $M$ trajectories, our goal is to associate each track with a candidate detection, spawn new tracks, or end a track (Figure 2(c)) in an online fashion.

We solve the data association problem by using a weighted bipartite matching algorithm. Affinities between tracks and new detections are calculated from two criteria: overlap between projections of current trajectories forward in time and bounding boxes candidates; and the similarity of the deep representation of the appearances of new and existing object detections. Each trajectory is projected forward in time using the estimated velocity of an object and camera ego-motion. Here, we assume that ego-motion is given by a sensor, like GPS, an accelerometer, gyro and/or IMU.

We define an affinity matrix $A(\tau_a, s_a)$ between the information of an existing track $\tau_a$ and a new candidate $s_a$ as a joint probability of appearance and location correlation.

$$A_{\text{deep}}(\tau_a, s_a) = \exp(-||F_{\tau_a}, F_{s_a}||_1) \quad (1)$$

$$A_{2D}(\tau_a, s_a) = \frac{d_{\tau_a} \cap d_{s_a}}{d_{\tau_a} \cup d_{s_a}} \quad (2)$$

$$A_{3D}(\tau_a, s_a) = \frac{M(X_{\tau_a}) \cap M(X_{s_a})}{M(X_{\tau_a}) \cup M(X_{s_a})} \quad (3)$$

where $F_{\tau_a}, F_{s_a}$ are the concatenation of appearance feature $f_{\text{app}}$, dimension $D$, center $c$, orientation $\theta$ and depth $d$. $X_{\tau_a}$ and $X_{s_a}$ are the tracked and predicted 3D bounding boxes, $M$ is the projection matrix casting the bounding box to image coordinates, $A_{2D}$ and $A_{3D}$ is the Intersection of Union (IoU).

$$A(\tau_a, s_a) = w_{\text{deep}} A_{\text{app}}(\tau_a, s_a) + w_{2D} A_{2D}(\tau_a, s_a) + w_{3D} A_{3D}(\tau_a, s_a) \quad (4)$$

$w_{\text{deep}}, w_{2D}, w_{3D}$ 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 [45].

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

**Depth-Ordering Matching.** We introduce instance depth ordering for assigning a detection to neighbor tracklets, which models the strong prior of relative depth ordering found in human perception. For each detection of interest (DOI), we consider potential associated tracklets in order of their depths. From the view of each DOI, we obtain the IOU of two non-occluded overlapping map from both ascending
and descending ordering. To cancel out the ordering ambiguity of a distant tracklet, we filter out those tracklets with a larger distance to a DOI than a possible matching length. So Equation 3 becomes

$$A_{3D}(r_a, s_a) = I \times \frac{\phi(M(X_{r_a})) \cap M(X_{s_a})}{\phi(M(X_{r_a})) \cup M(X_{s_a})},$$  

where $I$ denotes if the tracklets is kept after depth filtering, and the overlapping function

$$\phi(\cdot) = \arg \min_{x} \{x|\text{ord}(x) < \text{ord}(x_0) \forall x_0 \in M(X_{r_a})\}$$

captures pixels of non-occluded tracklets region with the nearest depth order. It naturally provides higher probabilities of linking neighbor tracklets than those layers away. In this way, we obtain the data association problem of moving objects with the help of 3D trajectories in world coordinates. Figure 3 depicts the pipeline of depth ordering. Finally, we solve data association using the Kuhn-Munkres algorithm.

**Occlusion-aware Data Association.** Similar to previous state-of-the-art methods [42, 43, 35], we model the lifespan of a tracker into four major subspaces in MDP state space: \{birth, tracked, lost, death\}. For each new set of detections, the tracker is updated using pairs with the highest affinities score (Equation 4). Each unmatched detection spawns a new tracklet; however, an unmatched tracklet is not immediately terminated, as tracklets can naturally disappear in occluded region and reappear later. We continue to predict the 3D location of unmatched tracklets until they disappear from our tracking range (e.g. -10m to 100m) or die out after 20 time-steps. We address the dynamic object inter-occlusion problem by separating a new state called “occluded” from a lost state. An object is considered occluded when covered by another object in the front with over 70% overlap. An occluded tracklet will not update its lifespan or its feature representation until it is clear from occlusion, but we still predict its 3D location using the estimated motion. Figure 4 illustrates how the occlusion-aware association works. 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. We embed a 3D location $P$ to a 64-dim location feature and use 128-dim hidden state LSTMs to keep track of a 3D location from the 64-dim output feature.

Prediction LSTM (P-LSTM) models dynamic object location in 3D coordinates by predicting object velocity from previously updated velocities $\hat{P}_{T-n:T-1}$ and the current possible location $\hat{P}_T$. We use previous $n = 5$ frames of vehicle velocity to model object motion and acceleration from the trajectory. Given the current expected location of the object from 3D estimation module, the Updating LSTM (U-LSTM) considers both current $\hat{P}_T$ and previously predicted location $\hat{P}_{T-1}$ to update the location and velocity (Figure 2(c)).

Modeling motion in 3D world coordinates naturally cancels out adverse effects of ego-motion, allowing our model to handle missed and occluded objects. The LSTMs continue to update the object state

$$s_a^{(i)} = s_{a-1}^{(i)} + \alpha(s_a^* - s_{a-1}^*)$$

using the observation of matched detection state $s_a^*$ with an updating ratio $\alpha = 1 - A_{deep}(\tau_a^*, s_a^*)$, while assuming a 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. 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 LSTM motion estimator updates the velocity and states of each object 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 a 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 [47]. 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 annotations at no cost of human efforts. Our data collection and annotation pipeline extend the previous works like VIPER [32] and FSV [20], especially in terms of linking identities across frames. Details on the thorough comparison to prior data collection efforts, and dataset statistics can be found in the supplementary material.

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, providing the potential of large-scale data collection, while VIPER requires expensive off-line processing.

5. Experiments

We evaluate our 3D detection and tracking pipeline on Argoverse Tracking benchmark [6], KITTI MOT benchmark [13] and our large-scale dataset, featuring real-world driving scenes and a wide variety of road conditions in a diverse virtual environment, respectively.

5.1. Training and Evaluation

Dataset. Our GTA 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. The KITTI Tracking benchmark provides real-world driving scenario. Our 3D tracking pipeline train on the whole training set and evaluate the performance on the public testing benchmark. The Argoverse Tracking benchmark offers novel 360 degree driving dataset. We train on the training set and evaluate the performance on the validation benchmark since the evaluation server is not available upon the time of submission.

Training Procedure. We train our 3D estimation network (Section 3.3) on each training set, separately. 3D estimation network produces feature maps as the input of ROIAlign [17]. The LSTM motion module (Section 3.5) is trained on the same set with a sequence of 10 images per batch. For GTA, all the parameters are searched using validation set with detection bounding boxes from Faster R-CNN. The training is conducted for 100 epochs using Adam optimizer with an 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. More training details can be found in supplementary material.

3D Estimation. We adapt depth evaluation metrics [10] from image-level to object-level, leveraging both error and accuracy metrics. Error metrics include absolute relative difference (Abs Rel), squared relative difference (Sq Rel), root mean square error (RMSE) and RMSE log. Accuracy metrics are percents of $y_i$ that $\max(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}) < \delta$ where $\delta = 1.25, 1.25^2, 1.25^3$. Following the setting of KITTI [13], we use orientation score (OS) for orientation evaluation.

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 $DS = \min(V_{\text{pred}} / V_{\text{gt}}, V_{\text{gt}} / V_{\text{pred}})$ with an upper bound 1, where $V$ is the volume of a 3D box by multiplying its dimension $h \times w \times l$. A Center Score ($CS$) measures distance of a projected 3D center and a ground truth. $CS$ is calculated by $CS = (1 + \cos(\alpha \Delta_x - \Delta_y)) / 2$, with a upper bound 1, where $\alpha$ is an angular distance $((x_{gt} - x_{pd}) / w_{pd}, (y_{gt} - y_{pd}) / h_{pd})$, weighted by corresponding box width and height in the image coordinates.

Object Tracking. We follow the metrics of CLEAR [2], including multiple object tracking accuracy (MOTA), multiple object tracking precision (MOTP), miss-match (MM), false positive (FP), and false negative (FN), etc.

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

5.2. Results

3D for tracking. The ablation study of tracking performance could be found in Table 1. Adding deep feature distinguishes two near-overlapping objects, our false negative (FN) rate drops with an observable margin. With depth-order
Table 1: Ablation study of tracking performance with different methods in GTA dataset. Motion column shows which motion model being used. KF stands for Kalman Filter. Ratio is the relative improvement in reducing the number of mismatch pairs. Using deep feature in correlation can reduce the false negative (FN) rate. Using depth-order matching and occlusion-aware association filter out relatively 6 – 8% mismatching trajectories. LSTM modeling dynamic motion benefits 3D IoU AP in Table 2.

Table 2: Comparison of tracking performance on 3D IoU AP 25, 50, 70 in GTA dataset. Noted that KF3D slightly improves the AP performance compare to single-frame estimation (None), while 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 LSTMs.

Table 3: Importance of using projection of 3D bounding box center estimation on KITTI training set. We evaluate our proposed model using different center inputs \( c \) to reveal the importance of estimating projection of a 3D center. The reduction of ID Switch (IDS), track fragmentation (FRAG), and the increase of MOT A suggest that the projection of a 3D center benefits our tracking pipeline over the 2D center.

**Figure 5:** Qualitative results of 3D Estimation on KITTI testing set. We show predicted 3D layout colored with tracking IDs.
Table 4: Performance of 3D box estimation. The evaluation demonstrates the effectiveness of our model from each separate metrics. The models are trained and tested on the same type of dataset, either GTA or KITTI. With different amounts of training data in our GTA dataset, the results suggest that large data capacity benefits the performance of a data-hungry network.

Table 5: Tracking performance on the validation set of Argoverse tracking benchmark [6]. Note that the LiDAR-centric baseline[6] uses LiDAR sweeps, HD maps for evaluation.

Table 6: Tracking performance on the testing set of KITTI tracking benchmark. Only published methods are reported. † sign means 3D information is used.

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 a 3D world. Our pipeline consists of a single-frame monocular 3D object inference model and motion LSTM for inter-frame object association and updating. In data association, we introduced occlusion-aware association to solve inter-object occlusion problem. In tracklet matching, depth ordering filters out distant candidates from a target. The LSTM motion estimator updates the velocity of each object independent of camera movement. Both qualitative and quantitative results indicate that our model takes advantage of 3D estimation leveraging our collection of dynamic 3D trajectories.

7. Acknowledgements

The authors gratefully acknowledge the support of Berkeley AI Research, Berkeley DeepDrive and MOST-107 2634-F-007-007, MOST Joint Research Center for AI Technology and All Vista Healthcare.
References


