Deep 3D Vehicle Tracking using Region-based LSTMs

Ji Lin[1] Dequan Wang[2] David Zhang[2] Philipp Krähenbühl[3] Trevor Darrell[2] Fisher Yu[2]
[1]Tsinghua University [2]UC Berkeley [3]UT Austin

Abstract

Contemporary single-frame CNN models and datasets have led to significant progress inferring 2D object properties in complex scenes, yet it is still a challenge to infer dynamic object relationships in real-world driving settings. We develop a novel method using a region-based recurrent tracking architecture that can obtain accurate 3D trajectories of moving vehicles from monocular dashcam videos. We base our model on a faster R-CNN front-end, which efficiently finds candidate region proposals and infers 2D target object positions. We further incorporate a recurrent network to track detected objects over time in 3D, using a LSTM model for each object box. To train our model we create a novel dataset of synthetic driving scenes suitable for learning object dynamics across a wide range of settings. Our dataset is an order of magnitude larger than existing synthetic driving datasets and includes temporal sequences suitable for learning tracking and dynamics. Our experiments confirm that our model leverages our larger and temporally structured dataset to predict smooth trajectories that are more accurate than those predicted by a single-frame model.

1. Introduction

Deep learning has achieved great progress in the area of self-driving cars, including methods for object detection and scene segmentation from monocular images. Many successful approaches to autonomous driving control rely on additional views or sensors, sensing or reconstructing 3D point clouds before inferring object trajectories in 3D. In contrast, human observers have no difficulty to perceive the 3D world (including object trajectories) from simple sequences of 2D images. In this paper we explore architectures and datasets sufficient to develop similar capabilities using deep convolutional networks.

We employ a base model for layout and configuration prediction which leverages a state-of-the-art model for object detection. We use that model to predict initial scene layouts, from which we instantiate inference and tracking modules for each detected object instance in the scene. A recurrent network architecture within each instance-specific network module integrates observations associated with the instance over time, improving the accuracy of estimated parameters. Temporal integration is especially valuable to stabilize inference of certain 3D parameters such as out of plane object rotation. Our modules are also conditioned on overall scene appearance, and therefore the model learns an implicit scene-conditional prior on these parameters.

Concretely, our model extends recent single-frame monocular object detection methods to include object-level pose inference and tracking using recurrent neural networks. We follow the R-CNN meta-algorithm, extending the region processing to include recurrent network tracking. We employ LSTMs for each object instance, and refer to our method as Region-based LSTMs (R-LSTMs).

Critical to achieving success with such a model is a dataset from which one can learn model parameters. No existing dataset is sufficient; while existing datasets with large-scale variation in appearance are available, they are limited to static scenes. Datasets with vehicle or object tracks exist, but they do not have the required annotations for training our model, are generally too small in scale to train a contemporary large deep model, and often do not exhibit significant variation in scene appearance and/or geometry. We collect a new dataset which does have these required elements, leveraging a realistic popular synthetic driving environment, augmented with novel instrumentation to collect previously unavailable dynamic metadata associated with the observed scenes.

We collect driving sequences from the well-known GTAV environment, and utilize several novel techniques to obtain instance-associated temporal trajectories (See Figure). We develop specific vertex and pixel shaders which can be injected into the rendering pipeline which allow us to collect post-hoc rendering metadata. Such metadata can be captured in associated texture buffers, and read out to access the internal structures utilized in the rendering pipeline, which reflect the underlying ("ground-truth") scene geometry. In contrast to previous approaches, which focused on static label cues, our approach also allows the collection of temporal instance and tracking data.

As we show below, this dataset is visually diverse and rich enough to capture a wide range of scene appearances

and dynamics. We experiment with our model in a variety of settings, inferring both object position and orientation, and improving estimates over time using temporal cues. Our model demonstrates dynamic vehicle situational awareness is available from simple monocular sequences which may be complementary to other sensors, and/or available on lower-cost or more widely deployed platforms than existing multi-sensor approaches.

We plan to release our model implementation, network weights, and the full dataset upon acceptance.

2. Related Works

3D Layout Estimation The problem of object 3D layout estimation is a 7 DoF problem, where we need to estimate the 3D translation, rotation and dimension of the object. Classical problem of object pose estimation from 2D image includes the estimation of rotation and translation, which was previous considered as a purely geometric problem named as perspective n-point problem (PnP). Previous approaches [18] assume correspondences between 2D key points and the 3D object model. Other methods [8, 25] construct 3D models and find the 3D pose that match the image best. Recent datasets [10, 31, 32, 19] has extended 3D pose estimation to object categories based estimation, requiring handling the variance from both pose and appearance changes. Tulsiani & Malik [30] introduced RCNN based detection framework to detect objects and passed resulting regions to a pose estimation network. Poirson *et al*. [22] trained a deep CNN to jointly perform 2D detection and view point estimation in a discretized space. Alahi *et al*. [1] uses LSTM to model interactions among persons, but our framework takes visual inputs directly instead of abstract representations.

Depth Estimation Estimating the 3D translation of objects is a 3 DoF problem, however, the 2D image coordinates will reduce 2 DoFs bounded by the projection rule. Estimating the wholes scene depth from single monocular image has been studied by previous works. Saxena *et al*. [27] predicted depth from a set of image features using linear regression and a MRF. Ladicky *et al*. [17] integrate semantic object labels with deep features to improve depth estimation performance. Eigen *et al*. [7] present a multi-scale deep network to predict the depth map from coarse to fine, which was extended to predict depth, surface normals and semantic labels at ones [6]. For specific driving scenarios, Neven *et al*. [21] proposed an efficient framework that jointly solved semantic segmentation, instance segmentation and depth estimation at runtime, where the depth of the vehicle is estimated by averaging over the semantic mask on depth map.

Driving Benchmarks There has been several datasets on driving scenarios, including real-world driving datasets KITTI [10], Cityscapes [5] and BDD dataset [33]. The KITTI benchmark suite provides well annotated ground truth for visual odometry, stereo reconstruction, optical flow, scene flow, object detection and tracking, supporting low-level and high-level vision tasks on the same data, while its size is limited. The Cityscapes benchmarks provides semantic segmentation and instance segmentation on driving data from 50 European cities. Despite its large size, only 5,000 frames is well annotated, highlighting the challenges of annotating real-world images. BDD benchmark provides large scale crowd-sourced video data to support end-to-end driving model.

The real-world datasets are very expensive to collect and label, so they are either small in scale or poorly annotated, especially the 3D information, which is hard to annotate by human and needs expensive LiDAR. To overcome this difficulty, there has been works on virtual driving datasets, like virtual KITTI [9], SYNTHIA [24] and VIPER [23], which have been proven to be very effective for learning driving. The most similar work to our dataset is VIPER, which also utilized GTA V as a simulation environment, providing 250k fully annotated high resolution images by injecting shaders. Nevertheless, the annotation in VIPER is more related to low level perceptual tasks like object detection, segmentation, optical flow estimation, *etc*., which is not fully suitable for studying planning. Also we show that we can easily benefit from larger data size than VIPER dataset in understanding 3D driving scenarios.

Tracking 2D tracking has been surveyed extensively in [34, 26, 28]. Some previous methods [4, 9, 16] track objects based on correlation filters, which regresses all the circular-shifted versions of input features to a Gaussian function, and is computational efficient in Fourier domain. More recent methods adopt CNN for visual tracking. Existing CNN trackers typically builds on pretrained object recognition networks. Some generic object trackers are trained entirely online, starting from the first frame of a given video [11, 2, 15]. 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, neural networks are usually slow to train, which will bring considerable burden at inference. And the online training methods cannot utilized the large amount of video data. Held *et al*. [14] proposed a regression based method for offline training of neural networks that can track novel objects at test-time at 100 fps. Siamese networks are also utilized for tracking, including tracking by object verification [29], tracking by correlation [3]. All those methods only takes 2D image visual features into consideration, where the search space is restricted near the original position of the object. Though such practice is acceptable for most cases, we show that if we can refine the search space with 3D information by re-projection into old camera

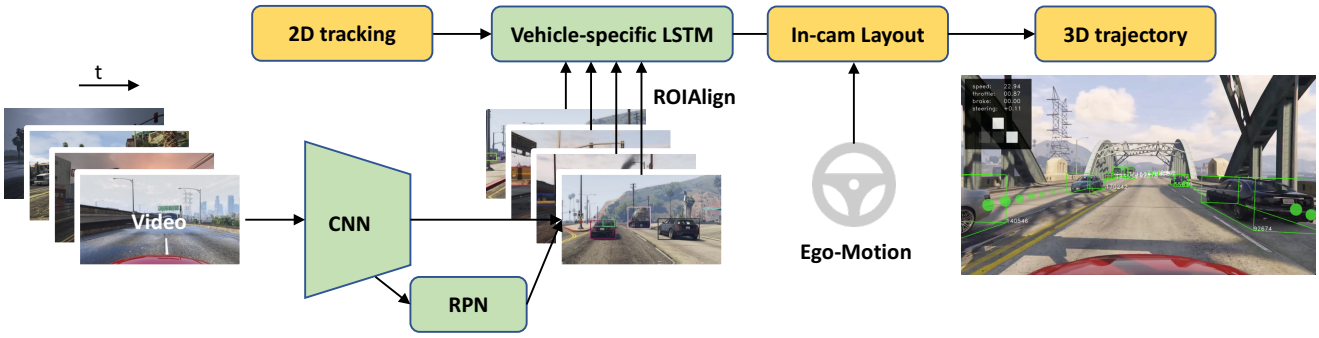

Figure 1. The overall view of our framework. To gain an understanding of the 3D driving scenario, we first obtain the 2D object detection result. Afterwards region-based features are extracted with ROIAlign technique and assigned with vehicle IDs from 2D tracking. The features are then fed into vehicles-specific LSTM to produce temporal consistent result in ego camera coordinate. Combined with the transition of camera extrinsic, we can reconstruct the 3D trajectories of other vehicles in world coordinate, which helps planning and future trajectory prediction.

coordinate, we can further reduce tracking error.

3. Region-based LSTMs

We develop a region-based approach to recurrent network-based tracking, following the widely adopted R-CNN meta-algorithm. We detect object regions and infer their properties in 2D using a single-frame model, and then instantiate LSTM modules for each inferred object region to perform tracking over time. The overall framework of our method is depicted in Figure 1.

3.1. Base Network Structure

To obtain a accurate understanding of the task and the dataset and exclude the effect of 2D detection, our experiments were conducted with ground-truth bounding boxes.

Raw images are input to the base CNN network, which computes a convolutional feature map. We used ROIAlign to get the object-specific feature given the detected ROIs. A multi-task head regresses to the size of an object D, its local rotation θ_l and inverse distance $\frac{1}{d}$ to the camera.

We directly regress to dimension D and depth d using fully connected layers and MSE loss. For the estimation of θ_l, we used MultiBin [20] method, where we first classified the alpha into 2 bins and then regressed the residual relative to the center of the bin. Compared to [20] where ROIs are cropped from images and fed into regression network, our method calculates whole scene feature map and thus utilizes the context information. We show that context information is important for 3D layout estimation. By integrating the detection network into our framework, we get the 3D estimation at almost no cost.

Given the detected ROIs, we estimated the 3D layout of the target objects (vehicles), including their translation in the camera coordinate and their dimension and rotation. The object dimension D has typically smaller variance since vehicles tend to be roughly the same size, and it is unrelated to rotation and translation while strongly related to the appearance of a certain subcategory of vehicle, like cars, vans, trucks, *etc*. Since we get a subcategory label from 2D detection result, we pre-calculated the mean dimension of each subtype c, and regress the residual $D - avg_c(D)$.

The second set of parameters to regress is the orientation around each axis (θ, ϕ, α). Similar to [10], we assumed that the rotation of vehicles is significant only around the vertical axis, so we only considered θ and assumed $\phi, \alpha = 0$. Estimating the global object rotation $R \in SO(3)$ in the frame only from local visual features is not possible, since the visual appearance is corresponding to the local rotation θ_l, which is the rotation with respect to the ray through the 2D bounding box center (as illustrated in [20]). We chose to regress the local rotation θ_l. Given the coordinate distance x to the vertical center of image and the focal length f, we can easily restore the global rotation θ in the camera coordinate from θ_l with simple geometery:

$$\theta = (\theta_l - \arctan\frac{x}{f}) \mod 2\pi$$

We followed the same axis setting as in [10]. The center of the vehicle bounding box is given, providing the direction of ray to the vehicle center, which reduces 2 DoFs, so we only need to regress 1 variable. Since the distance is most related to visual features, and the visual feature is related to the disparity of depth, we follow the usual convention and regress the inverse depth $\frac{1}{d}$. [1]

[1] To restore the coordinate (x, y, z), assume the coordinate of the bounding box in image plane with respect to the image center (that is camera plane) is (x_c, y_c) and the focal length is f, we can calculate: $x = \arctan(x_c/f) * d; y = \arctan(y_c/f) * d; z = d$.

3.2. Temporal Information Aggregation

The change of viewing angle, illumination, occlusion, *etc*. can bring trouble for 3D bounding box estimation. To exploit the temporal consistency of certain vehicles, after obtaining the tracking trajectory, we associate the information across frame by using a two-layer LSTM after ROI-pooling. The aggregated features are then fed into the multi-task head to produce the estimation of depth, rotation and dimension.

3.3. Tracking by Re-Projection

In real-life driving scenarios, we can easily get high frame-rate video inputs. High frame-rate videos greatly help tracking since we are able to exploit more locality across frames. Unlike normal tracking datasets where cameras are fixed or almost fixed, the tracking in driving will need to handle two difficulties, the ego-motion and the motion of other vehicles. The motion of other vehicles can be handled by exploiting locality, while the ego-motion will usually lead to much more dramatic location change, which is sometimes not possible to handle simply by locality, *e.g*. when we make a sudden turn to the right, everything on the image will shift left dramatically. As we have already estimated the 3D layout of other vehicles, and we can gain access to the change of camera extrinsic from IMU, we are able to re-project the locations of vehicles in the last frame to the current frame, which will help to offset the shift caused by ego-motion.

By using this technique to offset the change caused by ego-motion, we show that multi-vehicle tracking can be solved simply by exploiting locality in 15 or higher FPS scenarios. In experiments, given n_1 bounding boxes from last frame and n_2 bounding boxes in this frames, we computed the one-to-one IOU score resulting in a $n_1 \times n_2$ matrix. We set the IOU threshold = 0.1, by greedily finding the highest IOU matching pair and cross the corresponding row and column until all of the remaining IOU values are below threshold, we can obtain the association across two frames. With the re-projected bounding boxes from last frame as an accurate search space, we can further improve the accuracy by utilizing regression-based tracking framework like GOTURN [14]; we leave this as future work.

Training our network from scratch requires a considerable amount of training data; to obtain this, we utilize a simulated game environment, as described in the next section.

4. Simulator and dataset

We provide a novel large scale dataset for 3D driving scene understanding and high-level planning. The dataset includes annotations for 3D layout estimation, 3D tracking and prediction of future trajectories. We use Grand Theft Auto V, a modern game that simulates a functioning city and its surroundings in a photo-realistic three-dimensional world, to collect this data.

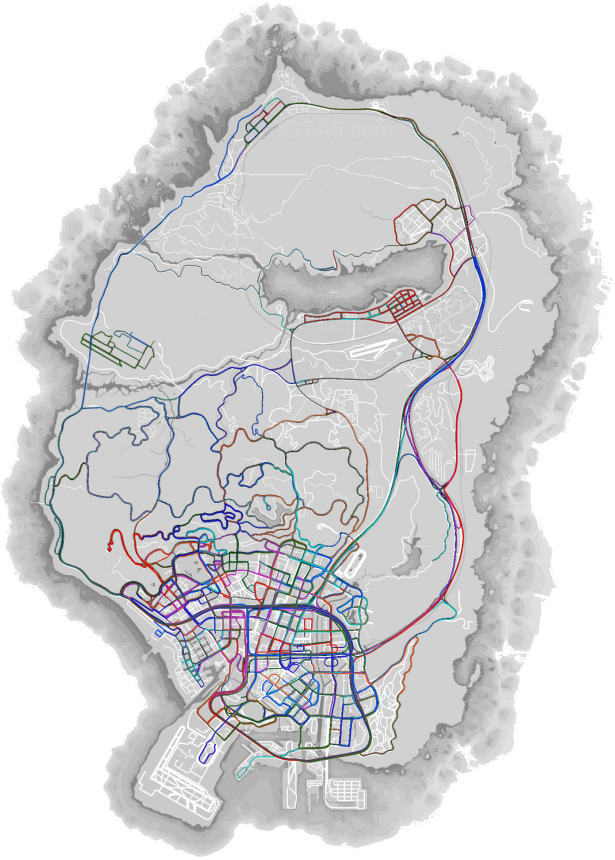

Figure 2. The road coverage of our data in GTA world. Each color represents a video clip. We sampled more data in urban area for higher data efficiency. Notice that some area is not visited since it is not accessible (like airport).

Our dataset is closely related to VIPER [23], while our focus lies on 3D understanding of driving scenes and higher level planning, rather than general semantic and instance segmentation. In our collection, all the data is collected when driving vehicles in the world. Our dataset is 5 times bigger that VIPER, providing 1.2M high resolution RGB images and corresponding depth map, which can be used to simulate a LiDAR sensor. We use the in-game API to capture low- and high-level annotations. In addition, we use shader injection to render out ground truth images, similar to Richter *et al*. [23]. In contrast, our data collection is real-time, while Richter *et al*. require expensive offline processing.

The real-time performance of our approach is one of the strongest aspects going forward. We don't just have a dataset with ground truth annotations, we provide a photo-

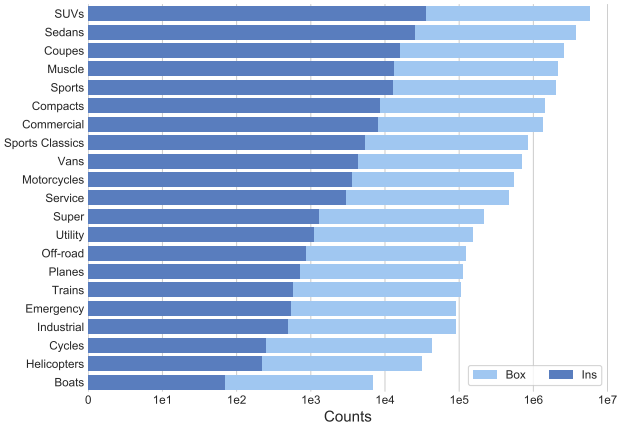

Figure 3. The number bounding boxes and unique instances of each fine-grained category.

realistic simulator with accompanying ground truth supervision for each frame. Instead of over-fitting to a fixed dataset, we can collect a very large amount of training data, especially in combination with reinforcement or imitation learning. The data can also be collected so as to adjusts to the driving policy that is currently being learned.

We provide dense 2D & 3D bounding boxes, 3D scene layout, motions (speed, direction, yaw rate of all the vehicles), trajectory information and ego-control signal like throttle, brake and steering. The sequence is recorded by an expert AI driver, written by the game designer and accessed through scripting engine. The AI follows all the traffic rules, stays in the correct lane, and stops at traffic lights. It could provide a good starting point for general planning and imitation learning. Our dataset features a full day-night cycle, and a diverse set of weather patterns including clear, extra sunny, rainy, thundering, smoggy, snowy, *etc*., and covers an extensive area of the game world. The road coverage of our data is shown in Figure 2. The vehicle diversity is also very large in GTA world, featuring 22 fine-grained subcategories, providing a benchmark for further study. We analyzed the distribution of the 22 fine-grained subcategories in Figure 3.

4.1. Hooking into GTA V

In order to interact with the GTA V engine, we use dll-injection to load custom code into the game, while it is running. Our code intercepts all DirectX 11 rendering commands and interfaces with the native scripting API of the game. In particular, we inject custom rendering code into the shaders of the game. In addition, we interface to the GTA script engine to query and modify the internal game state.

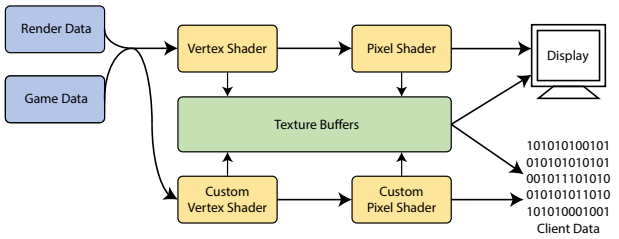

Figure 4. Visualization of the render pipeline after injecting custom shaders. We copy data to custom texture buffers to be retrieved at the end of the frame rendering

4.1.1 Shader Injection

We inject arbitrary code into the rendering pipeline of Grand Theft Auto V, by reverse engineering the DirectX 11 binary shader format. This allows us to replace, or modify any rendering code of the game. We modify the rendering pipeline to produce instance segmentation and depth outputs.

The rendering engine stores depth values inside a depth buffer. We directly read the depth buffer after the last object is rendered, and before any post-processing steps are applied. For numerical reasons, the GTA V engine internally stores the inverse depth. We compute the depth map by inverting the depth buffer and scaling it by 0.05 to convert from game units to meters.

For instance segmentation we follow an approach similar to Richter *et al*. [23]. We exploit the reference frame of objects drawn to group them into individual instances. All parts of a vehicle or person share the same reference frame, also known as *world transformation*. We extract this transformation matrix before each rendering call, and match them to the world position tracked by the native function API. Rendering calls with the same reference frame form an object. One exception are the wheels of vehicles which are animated separately in GTA V. Wheels are associated with their corresponding vehicle by taking advantage of the rendering order; wheels are always rendered immediately following the body of the vehicle.

In summary, shader injection allows us to extract both an instance segmentation, and a depth map in real time while the game is being played.

4.1.2 Native Function APIs

GTA V provides native functions to query and modify the inner workings of the game. We used *Script Hook V* library for scripting. Native functions allow for both data collection and set up driving scenarios. To build a realistic driving scenario, we attached the camera to the ego-vehicle, towards the forward direction of the vehicle. To increase

the diversity of the data, we randomly selected the starting point from a location pool, which was sampled proportional to the road length. The weather and time was also controlled through native functions. Furthermore, we adjusted the density of vehicle in game to simulate different traffic conditions, the ratio was set within range $0.8 - 1.4$ times of default density.

4.2. Analysis of 3D Layout Estimation

Our raw data is recorded at 15 FPS, which is helpful for temporal aggregation but too redundant for 3D layout estimation. Close frames are usually very similar and yield little change in the 3D layout. To strike a good balance between performance and training complexity, we sampled the videos at approximately 3 FPS. With the goal of autonomous driving in mind, we only care about vehicles closer than 100m, and also filtered out the bounding boxes smaller than 15 pixels. We removed video pieces where the player remains still for a long time period, like waiting for a red light. This further reduces redundancy. The dataset was split into train, validation and test set with ratio 2:1:1. We evaluated the performance of 3D layout estimation by calculating the Mean Absolute Error (MAE) of local rotation (divided by π), depth and the F-norm of dimension error. We also evaluated the Orientation Score (OS):

$$OS = \frac{\cos(\Delta\theta) + 1}{2} \tag{1}$$

We chose ResNet-50 [13] as backbone network. The first 4 stage of the network is stacked to produce the feature map with stride=16. ROIAlign [12] was then applied to the feature map to extract instance related features. The training was conducted for 20 epochs with initial learning rate 1e-3. No data augmentation was conducted. Since all the bounding boxes were evaluated, including the extreme occluded and truncated cases, the difficulty is higher than detected results.

4.2.1 Data Association and Post-Processing

The native function API and render engine are not perfectly synchronized, with a time delay of up to 10ms. However, this time delay is not severe, and a greedy association based in image location and depth works in all cases.

To match data between the rendering pipeline and the function API we compute two independent measures of depth and a 2D bounding box location. We then greedily match these two estimation. We used the instance segmentation map to cluster valid vehicles, and fit a tight 2D bounding box around each segment as detection ground-truth. In addition we compute the average depth for each object using the instance segmentation. The native function API provides us with 3D bounding boxes in world coordinate and camera matrices. We project the 3D bounding box into the image plane and to obtain a slightly looser 2D bounding box. An second depth measurement is obtained from native function API, measuring the distance between the camera and the world coordinate of object center. With two sets of bounding box and depth pairs, we greedily matched the objects from two domains by matching the minimum error.

Another challenge is that the native function API returns both visible, fully or partially occluded objects. Fully occluded objects are easily filtered out, as they do not appear in the instance segmentation map. We have two criterion for judging partial visibility, the ratio of valid pixel in the tight bounding box and the area ratio of tight bounding box to the loose one. The first criteria filters out objects where many center pixels are occluded, while the latter focuses on objects occluded from side. We label any object as partially occluded if any of the ratios falls below a threshold $t = 0.3$.

5. Experiments

Number of Training Data The VIPER dataset has 134K frames for training, but their ground-truth for 3D layout estimation is not available yet, we evaluated the data efficiency on our dataset, as the collection procedure of two datasets are quite similar. The video length of our training data is 400K frames, and we tested the performance using $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ and full data to show how we can benefit from more data. The results are shown in Table 5. We see a consistent trend of performance improvement as the amount of data increases. The performance of our algorithm has not plateaued yet, and collecting more free data could see it perform even better.

Context Information We evaluated how the performance is related to context information by comparing the cropped baseline, where the ROI is cropped from the original image resized as 224×224 and fed into a neural network with a multi-task head. We chose the same backbone network and multi-task head structure, where the extracted feature map is pooled to produce one-dimensional feature. The training is done with exactly the same protocol, and the comparison is shown in Table 3.

Since every cropped bounding box occupies the whole network, it will benefit from the high resolution feature. So the local rotation and dimension can be precisely estimated. However, since context information is lost during cropping, it will not be able to estimate depth accurately, which is context dependent.

Performance vs. Depth Similar to [21], we evaluated how performance changes with respect to depth. We evaluated mean absolute error (MAE) and absolute relative error (ARD) of depth as well as OS. The results are shown in Table 1. We can see a clearly trend that the layout of near vehicles is easier to estimate, which is important for autonomous driving. The distribution of predicted depth

Depth	MAE	ARD	OS
<100m	3.895m	9.757%	0.9536
<50m	2.667m	8.988%	0.9625
<25m	1.255m	7.968%	0.9700

Table 1. Performance of 3D layout estimation with different amount of data. The OS is higher the better.

	Rotation	Depth	Dimension	OS
Cropped	0.0721	4.1261	0.6665	0.9533
Context	0.0715	3.8952	0.6480	0.9536

Table 3. Comparison between our context-aware method and the cropped image baseline.

Amount	Rotation	Depth	Dimension	OS
1/4	0.1220	5.3531	0.8421	0.9058
1/2	0.0856	4.2566	0.7201	0.9369
3/4	0.0736	3.8810	0.6564	0.9483
1	0.0715	3.8952	0.6480	0.9536

Table 5. Performance of 3D layout estimation with different amount of data. Lower is better, except for OS.

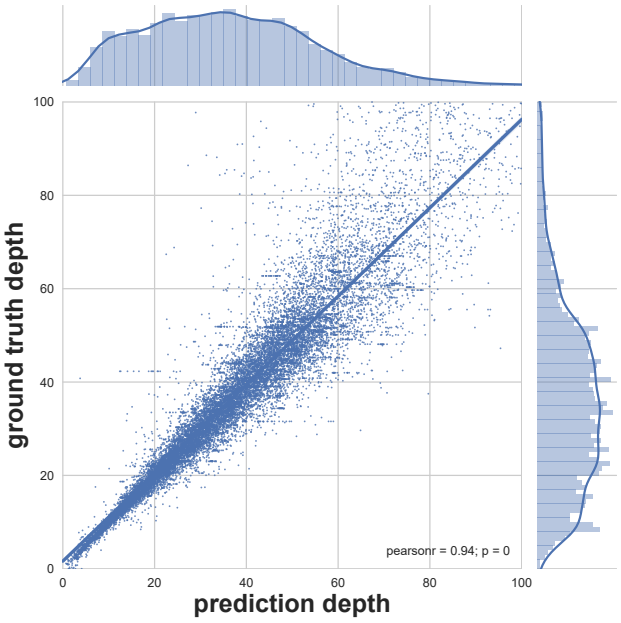

Figure 5. Distribution of predicted depth with respect to ground-truth depth.

with respect to ground-truth depth is provided in Figure 5. Again close predictions are accurate, while farther ones incur a larger estimation error.

Method	Accuracy	IOU
Vanilla	97.99%	0.8019
Re-Projection (estimated depth)	99.67%	0.8674
Re-Projection (gt depth)	99.67%	0.8683

Table 2. Comparison of tracking performance between vanilla IOU tracking result and re-projection technique.

	Rotation	Depth	Dimension	OS
Single	0.0715	3.8952	0.6480	0.9536
LSTM	0.0596	3.7862	0.7571	0.9658

Table 4. Temporal information aggregation validation.

5.1. Tracking by Re-Projection

In high-frame rate videos, given the detection results, tracking can be solved by exploiting locality between consecutive frames. With higher frame rate, the difference between two consecutive frames is smaller, and thus will provide better tracking result. In our dataset with 15 FPS videos, we show that we can handle most of the problem caused by other objects' motion by simply computing IOUs. However, since the difference caused by ego-motion is much more dramatic, especially when the vehicle turns left or right, where some of the bounding boxes between two consecutive frames may even separate with each other, with $IOU = 0$, for example in Figure 7. To address such scenarios, we used the re-projection technique mentioned above, where the depth is estimated using our previous trained depth estimation network.

At each frame, a bounding box can be assigned to a box in last frame, or be considered a new object. For each pair of bounding boxes between the current and last frame, we compute their overlap, and greedily assign each box in the current frame to the highest overlapping previous box. No box can be assigned twice. The result is provided in Table 2. To provide a theoretical upper bound, we first did an experiment with ground-truth data and found that our re-projection method eliminates 83.5% of tracking errors. We then replaced the depth with the estimate by our network and achieved similar performance. The re-projection technique is tolerant to the errors in the depth estimation and generally performs well.

5.2. Temporal Information Aggregation

Finally, 4 shows how the tracking can further be improved by aggregating the temporal information in a LSTM. The 3D pose estimated densely from frame to frame is not temporally consistent, which results in unstable trajectories. The LSTM based aggregation greatly smoothes the trajectory and reduce tracking errors.

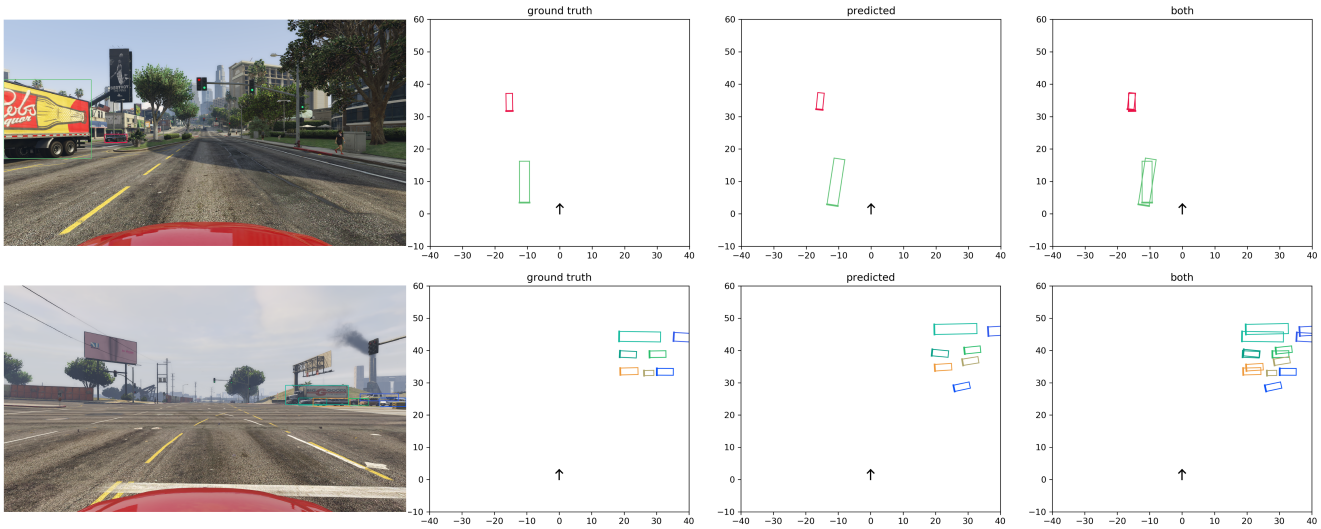

Figure 6. Birdview of 3D Pose Estimation. The first picture shows a relative easier case our method obtains high accuracy in depth, rotation and dimension. For the second difficult case where vehicles are extremely occluded while our frameworks successfully produce reasonable results.

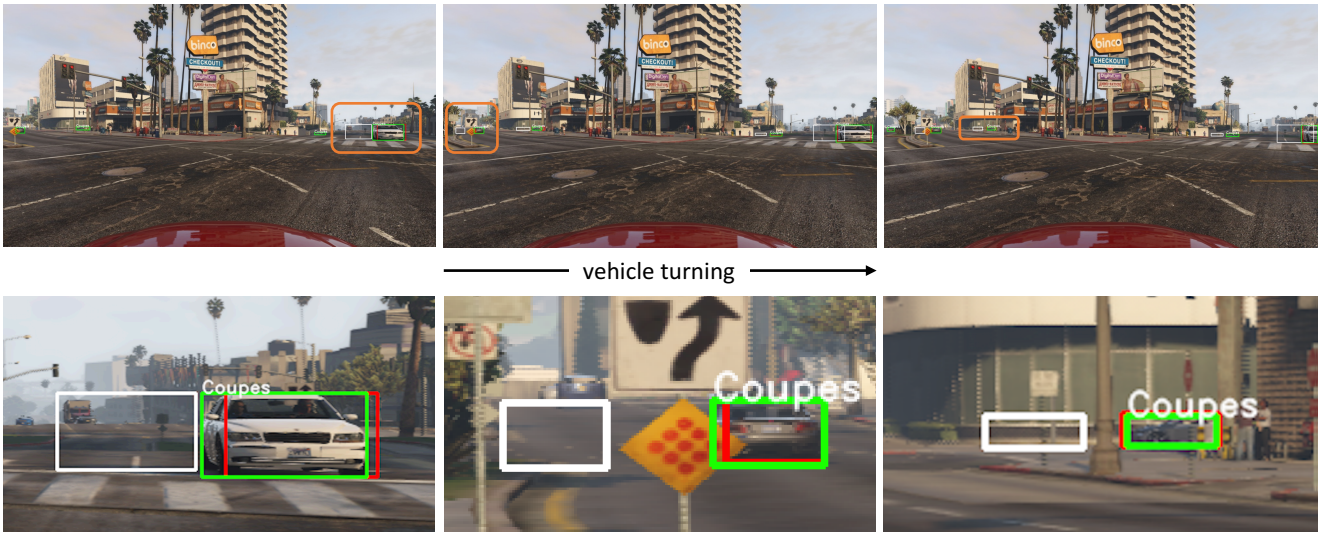

Figure 7. Example of cases where re-projection reduces error. The second row is the zoom in view of corresponding orange area in first row. The green bounding boxes are the detection result in current frame, and the white bounding boxes are the detection result from last frame. The white bounding boxes are refined with re-projection technique to produce the red bounding boxes as search area. Re-projection technique successfully addresses the bounding box shift caused by ego-motion and produces accurate search space in current frame.

6. Conclusion

In this paper, we go beyond infering 2D object properties in complex scenes and how to learn 3D vehicle dynamics from monocular videos. We propose a novel model, R-LSTM, combining image spatial region and temporal information to infer 3D trajectories of moving vehicles. We obtain a large-scale training dataset of synthetic driving scenes based on GTA V. Our collected data includes both control signal and dense label maps, and provides accurate vehicle annotations fully automatically. Our experiments show that our model extracts more 3D information from each image and the LSTM predicts more accurate trajectories based its internal temporal state representation.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in

crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016. 2

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009. 2

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pages 850–865. Springer, 2016. 2

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2544–2550. IEEE, 2010. 2

[5] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[6] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015. 2

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 2

[8] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67(2):159–188, 2006. 2

[9] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016. 2

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 3

[11] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2016. 2

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017. 6

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[14] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference Computer Vision (ECCV)*, 2016. 2, 4

[15] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012. 2

[16] 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 *Proceedings of the IEEE international conference on computer vision workshops*, pages 1–23, 2015. 2

[17] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–96, 2014. 2

[18] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009. 2

[19] K. Matzen and N. Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 761–768, 2013. 2

[20] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. *arXiv preprint arXiv:1612.00496*, 2016. 3

[21] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Fast scene understanding for autonomous driving. *arXiv preprint arXiv:1708.02550*, 2017. 2, 6

[22] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. Fast single shot detection and pose estimation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 676–684. IEEE, 2016. 2

[23] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *International Conference on Computer Vision (ICCV)*, 2017. 2, 4, 5

[24] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016. 2

[25] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, 2006. 2

[26] S. Salti, A. Cavallaro, and L. Di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Transactions on Image Processing*, 21(10):4334–4348, 2012. 2

[27] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006. 2

[28] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014. 2

[29] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2016. 2

[30] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015. 2

[31] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference on Computer Vision*, pages 160–176. Springer, 2016. 2

[32] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 75–82. IEEE, 2014. 2

[33] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint arXiv:1612.01079*, 2016. 2

[34] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006. 2