DeepAutoTrack (DAT): Vehicle Trajectory Prediction for Autonomous Driving

Anonymous CVPR submission

Paper ID 3790

Abstract

Vision-based deep neural networks are crucial components for autonomous driving in terms of visual understanding and decision making. These models need to be temporally consistent and visually interpretable. Most recent works focus on using static images for understanding visual semantics, ignoring the temporal consistency of driving conditions; or adopt end-to-end architectures for learning the pilot networks, providing limited interpretability. In this paper, we raise the question of "can we predict the car's future odometry given previous egomotion visual input?". The proposed system stems from the issue of human driver reaction time in the autonomous driving context. Our dual-staged model firstly applies modern convolutional object detectors for spotting traffic participants (i.e., cars in this paper) and robustly tracks the targets of interest. Then, a novel SEG-LSTM network is incorporated to fuse the multiple-streams from past frames and then predict targets' future trajectories. We demonstrate the feasibility of predicting future trajectories of vehicles for assisting mediated perception. We also show the effectiveness of combining various levels of abstractions(e.g., scene semantics, detection bounding boxes) further boost the prediction accuracy across diverse traffic conditions.

1. Introduction

Currently, there are two major paradigms for autonomous driving systems built upon vision-based input [6]: mediated perception approaches that firstly explain the vision input and then parse the scene to make a driving policy (usually by a controller with if-then-else rules); and behaviour reflex approaches that directly map the vision input to a driving policy by a regressor. In this paper, in the framework of first paradigm, we try to tackle the problem of predicting future trajectories of vehicle given its past information.

Being able to predict other traffic participants' future trajectories is important because it can help to prevent self-driving car from running into other cars. We need to know not just where other cars are, as in the localization case, but also how fast they are moving in order to avoid collisions. For a human driver, reaction time is a crucial factor that includes recognizing the light has changed, deciding to continue or brake, and whether to stop engaging the brake (remove foot from accelerator and apply brake). Accident reconstruction specialists commonly use 1.5 seconds [34]. Therefore, the ability to predict traffic participants' *future* trajectories will greatly benefit the mediated perception approach's driving policy decision making.

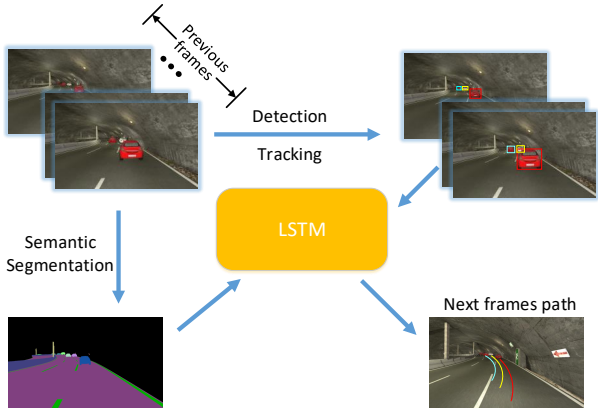


Figure 1: We propose a system to predict future trajectories of vehicles. Given historical information, obtained by detection, tracking and semantic segmentation, a multi-stream Long Short Term Memory (LSTM) based network is trained to generate future trajectory.

Nonetheless, vehicles trajectory prediction is a challenging problem: by dissecting visual scene understanding individually (*e.g.,* object detection, semantic segmentation, instance segmentation), there are still many open challenges for computer vision community. In addition, the scene parsing results contain quite some spurious information. Existing methods for scene parsing also mainly focus on static images. However, autonomous driving is intrinsically a dynamic problem. Human drivers make decisions by a sequence of input frames instead of one static frame. Therefore, temporal consistency among continuous frames

should also be taken into consideration when designing the interpretable system.

Vehicle trajectory prediction is also very related to visual tracking. In traditional vision based tracking problems (*e.g.*, [47, 36]) where target objects go through rather sporadic movements (usually the object trajectory is generated by artificial movement in order to test the robustness of the tracker), the prediction of the target's future trajectory is simply a moot problem. However, traffic participants generally exhibit regular trajectories by obeying a large number of common sense rulesp (*e.g.,* cars driving on the road, pedestrians walking on the sidewalks). Given law-abiding traffic participants, human drivers subconsciously project visual target's future trajectory. Nonetheless, due to the missing bridge between the traditional visual tracking community and the current autonomous driving research community, there is a lack of proper metrics for evaluating the temporal prediction result.

We address this issue through building a system that takes advantage of multiple streams with various levels of abstraction. In our approach, there are three key components: traffic participants detection, instance tracking and future trajectory prediction as shown in Fig. 1.Our main contributions are as follows:

- We construct a time-series dataset for vehicle trajectory prediction based on the SYNTHIA dataset [40][1] and verify the feasibility of predicting vehicles' future trajectories for assisting mediated perception.
- We take advantage of various levels of abstractions from multiple streams via recurrent network based temporal models and demonstrate the effectiveness of using "auxiliary information"(*e.g.*, scene semantics) further boost the prediction accuracy.
- Preliminary results by incorporating "social" factor demonstrate the potential of learning common sense rules with neighboring traffic participants in complex real world environment.

2. Related Work

Traffic participants detection Traffic participants detection is one key component of an autonomous driving system. Vehicle detection has been well studied for static highway mounted cameras: [22] present a concise overview of image processing methods and analysis tools before the "deep learning eras". A lot of progress has been made in recent years on object detection due to the use of convolutional neural networks (CNNs). Modern object detectors [20, 19, 39, 24, 23, 38, 32] are now good enough to be deployed in a consumer products. Faster R-CNN [39] proposes a region proposal network (RPN) and integrates RPN and Fast R-CNN [19] to build an end-to-end architecture. SSD [32] similarly uses a series of boxes and directly gives the probability for each object class.

Typical algorithms output bounding boxes on detected objects for a static image. And the deep neural nets can be transferred and fine-tuned to another set of classes for traffic participants we actually care about in the context of autonomous driving.

Instance association Instance association between frames is a prerequisite to solve the proposed intrinsically temporal problems. 3D scene flow is estimated in [3] by exploiting recognition to overcome the challenging scenarios in the presence of large displacement or local ambiguities. However, a total runtime of 10 minutes per frame is computationally prohibitive for the real-time deployment. To accomplish the task of instance association between frames, we resort to the well studied field of visual tracking. Discriminative Correlation Filters (DCF) [26] have demonstrated excellent performance for high-speed generic visual object tracking. Built upon their seminal work, there has been a plethora of recent improvements [28, 33, 12, 25, 45, 10] relying on convolutional neural network (CNN) pretrained on ImageNet as a feature extractor for visual tracking.

Most of the modern trackers focus on the problem of "class-agnostic" generic object tracking. In order to adapt to temporal changes, a continuous learning strategy is applied, where the model is updated rigorously in every frame. Such update is excessive and sensitive to sudden changed caused by, *e.g.*, scale variations, deformations, and out-of-plane rotations. This excessive update strategy cause both lower frame-rates and degradation of robustness due to over-fitting to the recent frames [10]. In the context of autonomous driving, however, we know in advance the classes of object of interests for tracking (*e.g.*, cars, pedestrians, cyclists). Therefore, with this extra source of information, we propose to use the neural nets object detection result as a more robust template to update the tacker's model to overcome the problem of model drifting. Hence the tracker is served solely as instance associator for the traffic participant.

Activity forecasting Activity forecasting is the task of predicting the motion or the action to be carried out by objects in a video. A body of work [5, 8, 15] explores various hierarchical models in face of different quality of data representation. Recently recurrent neural networks (RNN) and their variants including Long Short Term Memory (LSTM) [27] and Gated Recurrent Units (GRU) [7] have proven to be very successful for sequence prediction tasks. An action-conditioned video prediction model [17] that explicitly models pixel motion, by predicting a distribution over pixel motion from previous frames is proposed to learn from unsupervised video data. A curriculum learn-

[1] Dataset will be released upon paper publication.

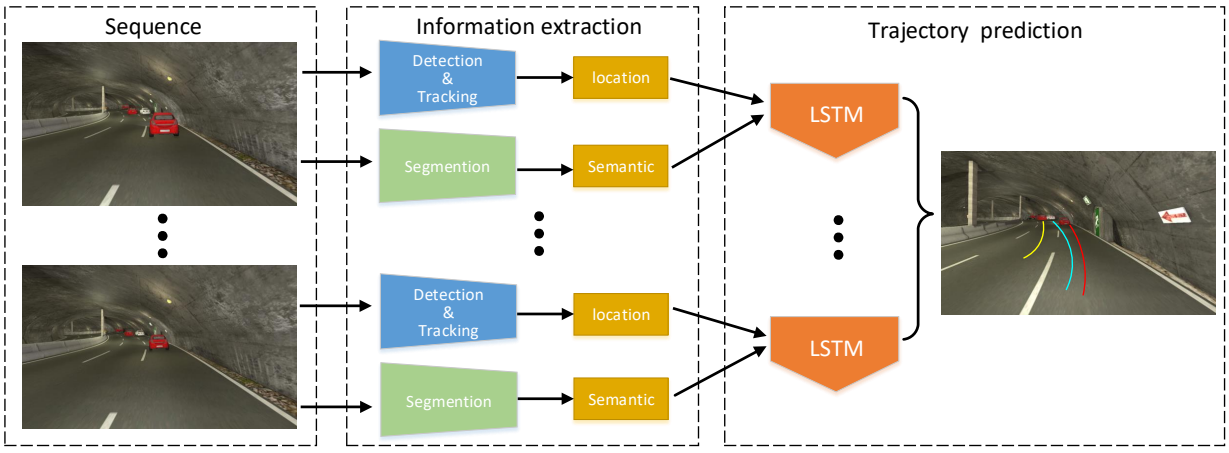


Figure 2: The framework for the proposed vehicle future trajectory prediction system. The input is the sequential data (RGB and/or depth maps) up to the current time step obtained from the host car. Target vehicles' sequential bounding boxes are extracted by detection & tracking module. Along with scene semantics as an auxiliary information, multi-stream inputs are fed into an LSTM to predict future trajectories taking advantages of various levels of abstractions.

ing strategy is proposed in [4] to gently change the training process from a fully guided scheme using the true previous token, towards a less guided scheme which mostly uses the generated token instead. A combination of a fully-convolutional network and an LSTM is proposed in [48] to learn from large-scale vehicle action data and is trained as a pilot network in the context of behaviour reflex approaches. Their goal is to predict future egomotion including multi-modal discrete and continuous driving behaviors. Inverse turning radius is predicted by a LSTM network in [31]: a heat map of attention is generated at each time step conditioned on the previous hidden states as a more succinct visual explanations.

More recent works also attempt to take the "social factor" into consideration. Social-LSTM [2] focus on modeling dynamics crowd interactions for path prediction by introducing a "social tensor" from a spatial-temporal neighborhood. Based on the social-LSTM, an attention mechanism is introduced in [44] built upon the high-level spatio-temporal graphs proposed in [30].

Datasets for autonomous driving Vision-based semantic segmentation in urban scene is a key functionality for autonomous driving. KITTI benchmark suite [18] provides a large amount of images of urban scene from Karlsruhe, Germany, with ground truth data for odometry, object detection, tracking, scene flow and stereo benchmarks. However, a limited 430 labelled images are provided for semantic segmentation. More recently, larger projects are constructed: Cityscapes dataset [9] which consists of a collection of images acquired in 50 cities around Germany, Switzerland and France in difference seasons, and having 5,000 images with fine annotations and 20,000 with coarse annotations over a total of 30 classes. Comma.ai [41] provides 7 and a quarter hours of largely highway driving. Udacity [1] dataset includes 65,000 labels across 9,423 frames at full resolution of 1920×1200 at 2Hz. The use of synthetic data has increased considerably in recent years within computer vision community. CARLA [14] and Sim4CV [35] are two recent endeavours that take advantage of simulated environment to support development, training and validation of autonomous urban driving systems. A synthetic dataset named SYNTHIA [36] is collected for investigating how useful synthetic images can be for semantic segmentation. Synthetic dataset has an obvious advantage: a fine annotated image in Cityscape dataset requires on average 1.5 hours which is very labor intensive. Thus, the cost of scaling large project would require a prohibitive economic investment in order to capture images from a larger variety of countries, in different seasons and different traffic conditions.

3. Proposed Algorithm

We first describe the overall framework for vehicle trajectory prediction and the details of implementation are presented in Sec 4.2. Our goal is via collecting visual information from the past frames to predict the traffic participants future trajectory. Fig. 2 shows the overall architecture of the proposed framework.

3.1. Problem formulation

We propose to learn a generic approach from history information and formulate the problem as predicting future traffic participants' trajectories. Formally, our problem can be defined as a mapping $\mathcal{M}$ between historical trajectory $\mathcal{J}_{\mathcal{H}} = \{j_p^t\}, t = 1, \ldots, T$ with auxiliary information

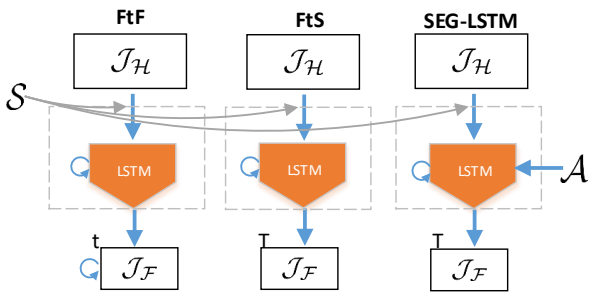


Figure 3: Illustrations of three temporal networks. $\mathcal{J}_\mathcal{H}$ represents historical trajectories, $\mathcal{J}_\mathcal{F}$ is the T time step future trajectories to be predicted. Left: Frame-to-Frame (FtF); Middle: Frame-to-Sequence (FtS); Right: SEG-LSTM, with auxiliary information as input $\mathcal{A}$. The social tensor $\mathcal{S}$ can be inserted into existing recurrent networks .

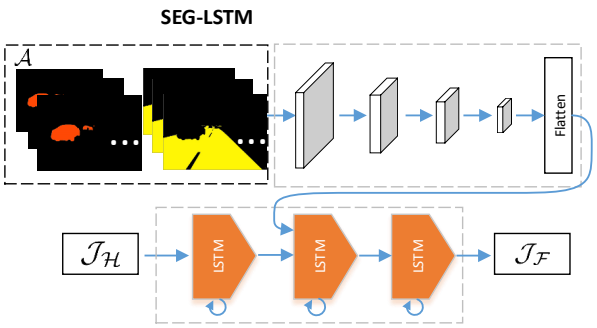


Figure 4: Diagram of SEG-LSTM model: semantic inputs are one-hot encoded images with channel number $\mathcal{C}$ as the number of semantic categories (we used only "car" and "road" as the two semantics used under the context of high-way driving). The higher level semantics are extracted through a four-layer CNN, producing a joint representation of the auxiliary semantic information and the co-ordinate trajectory information.

$\mathcal{A} = \{a_p^t\}, t = 1, \ldots, T$ for traffic participant p and the future trajectory $\mathcal{J}_\mathcal{F} = \{j_p^t\}, t = T+1, \ldots, T_F$:

$$\mathcal{J}_\mathcal{F} \leftarrow \boldsymbol{\mathcal{M}}_p(j, a) : \mathcal{J}_\mathcal{H} \times \mathcal{A} \tag{1}$$

where the traffic participants p can be vehicles, cyclists, pedestrians, *etc*. In this paper, we focus on vehicle as the sole traffic participant category and ignore the subscript p from now on. T_F is the length of future trajectory to be predicted (corresponding to the total driver reaction time). Historical and future trajectories are defined in a 3D occupancy grid:

$$\mathcal{J}_\mathcal{H}, \mathcal{J}_\mathcal{F} \in \Re^6 = \{x, y, w, h, d_{min}, d_{max}\} \tag{2}$$

where $\{x, y, w, h\}$ define target's 2D boundingbox in image pixel space and $\{d_{min}, d_{max}\}$ define the minimum and maximum relative distance between host car and the tracking target (acquired from the depth camera). Given RGB cameras' focal length f, the pixel space 2D boundingbox can be projected into 3D real-world coordinates $\{x_r, y_r, w_r, h_r\}$ as:

$$x_r = \frac{x}{f}d, y_r = \frac{y}{f}d \tag{3}$$

$$w_r = \frac{w}{f}d, h_r = \frac{h}{f}d \tag{4}$$

where d is distance between the center of the camera lens and the center of target. We use the minimum relative distance d_{min} as a close approximation.

3.2. Sequence modeling

For predicting the vehicle's future trajectory, we compare three temporal models based on LSTM cells. A "social pooling tensor" that implicitly reasons about the motion of neighboring traffic participants can be inserted into existing recurrent network architectures without any extra training supervision or modification to the optimization process as shown in Fig. 3. The first two models take sequential 3D occupancy as input. In the third model, we explore using semantic segmentation as a side "auxiliary" information $\mathcal{A}$ to better guide the trajectory prediction.

Frame-to-Frame (FtF) model: is akin to sequence generation model from Graves [21]. At test-time t, it samples current predicted 3D occupancy as feedback to the model for predicting next frame occupancy grid. The model is frame to frame Markovian and the next frame trajectory j_{t+1} can be estimated as:

$$j_{t+1} \leftarrow \boldsymbol{\mathcal{M}}(j_t, j_H) : \mathcal{J}_\mathcal{H} \tag{5}$$

Frame-to-Sequence (FtS) model: differs from the FtF model in that instead of sampling frame by frame for predicting future trajectories, FtS model directly predicts a trajectory sequence of length. As it pointed out in [4] that FtF training paradigm, predicting one token at a time, conditioned on the state and the previous correct token, is different from how we actually use them during inference and thus is prone to the accumulation of errors along the decision paths. Therefore, a curriculum learning approach is proposed to slowly change the training objective from an easy task, where the previous token is known, to a realistic one, where it is provided by the model itself. The aforementioned learning strategy applies to inference for time-varying sequence where an <EOS>token is needed to signify the end of the sequence. For our continuous prediction task, we stipulate the length of the prediction is predefined as T_F. Hence we have the assumption that current state is complete, in the sense it contains all historical information

required for predicting the future trajectory of the vehicle. The whole trajectory $j_{t+1,...,T_F}$ to be predicted can be estimated as:

$$j_{t+1,...,T_F}, \leftarrow \boldsymbol{\mathcal{M}}(j_H) : \mathcal{J}_{\mathcal{H}} \tag{6}$$

The whole sequence inference paradigm is more stable than the first FtF approach because of the elimination of the dependency of previous token. Such dependency will introduce the discrepancy between how the model is used at training and inference. Inference with the whole sequence will generate smoother predictions.

SEG-LSTM model: takes advantages of scene semantics from historical frames as auxiliary information $\mathcal{A}$. The second LSTM layer fuses the output from the bottom-most LSTM that encodes co-ordinates embedding with higher level semantic information. With auxiliary $\mathcal{A}$, the trajectory $j_{t+1,...,T_F}$ to be predicted can be estimated as:

$$j_{t+1,...,T_F}, \leftarrow \boldsymbol{\mathcal{M}}(j_H, a) : \mathcal{J}_{\mathcal{H}} \times \mathcal{A} \tag{7}$$

Semantic inputs are one-hot encoded images with channel number $\mathcal{C}$ as the number of semantic categories as in Fig. 4. The higher level semantic is extracted through a four-layer CNN, producing a joint representation of the semantic auxiliary information and the co-ordinate trajectory information. As described above, the 3D occupancy input has only six dimensions: $[x, y, w, h, d_{min}, d_{max}]$ encoding the co-ordinates information. So the bottom-most LSTM not only can learn the temporal information between time-varying trajectories, but also can balance input dimensions of both co-ordinate information and the semantic information.

Note that this model differs from the way semantic information is used in [48]: instead of using semantic segmentation as a "privilege information" which is missing at test time, SEG-LSTM directly takes historical scene semantics as input and directs the recurrent net future trajectory prediction.

Social tensor: was firstly introduced in [2] by observing the "common sense" rules and social conventions from pedestrians in crowded public space. Naturally, in the context of autonomous driving, such observation is also complied: when there are multiple cars occupying on-going lanes, the social interactions between traffic participants confine the space for their possible trajectories. The social interactions can be encoded implicitly by using a social pooling layer: at every time-step, the LSTM cell receives pooled hidden-state information from the LSTM cells of neighbors. While pooling the information, the spatial information is preserved by grid based pooling. The hidden state h_i^t of the LSTM at time t captures the latent representation of the i^{th} target in the scene. This representation can be shared with neighbors by building a social hidden tensor s_t^i:

$$s_t^i = \sum_{j \in \mathcal{N}_i} \mathbf{1}_{m,n}[x_t^j - x_t^i, y_t^j - y_t^i] \centerdot h_{t-1}^j \tag{8}$$

where $\mathbf{1}_{m,n}[x_t^j - x_t^i, y_t^j - y_t^i]$ is an indicator function to check if (x, y) is in the (m, n) cell of the grid, and $\mathcal{N}_i$ is the set of neighbors corresponding to target i. The trajectory to be predicted can be estimated as:

$$j_{t+1,...,T_F}, \leftarrow \boldsymbol{\mathcal{M}}(j_H, s_H^i) : \mathcal{J}_{\mathcal{H}} \times \mathcal{S} \tag{9}$$

This social pooling layer does not introduce any additional parameters and can be inserted in to the hidden layer of LSTM and training can be formulated as a neighboring batch update scheme. An important distinction from the vanilla LSTM is that the hidden states of multiple LSTMs are coupled by the social pooling layer. At every time-step, the update are jointly back-propagated through multiple LSTMs in a scene.

4. Experiments

We first provide the data collection procedure for vehicle trajectory prediction. And then we present the implementation details with performance analysis of individual modules. Quantitatively examination of SEG-LSTM architecture is also presented showing the advantages of using semantics directly as input can better assist the neural net's trajectory prediction.

4.1. Dataset collection

Currently, most public datasets with semantic labeling are single frame, static images. Continuous video streams are required for the purpose of car trajectory prediction. We collect a car trajectory prediction dataset based on the SYNTHIA [40] dataset. SYNTHIA dataset is a large corpus of synthetic images originally collected for the purpose of semantic segmentation of urban scenes generated by rendering a virtual city created with the Unity development platform. The potential of this virtual world includes extension capabilities: new parts of the cities and road conditions can be easily generated by adding different setups. The major features of the collected dataset include: scene diversity (European style town, modern city, highway and green areas), variety of dynamic objects (cars, pedestrians and cyclists), multiple seasons (dedicated themes for winter, fall, spring and summer), lighting conditions and weather (dynamic lights and shadows, several day-time modes, rain mode and night mode). There are more than 200,000 HD (760×1280) photo-realistic frames from video streams. Frames are acquired from multiple view-points (up to eight views per location), and each of the frames also contains an associated depth map. In our experiment, we used only left front camera view data.

Ground truth collection for car trajectory prediction

Step 1 → acquiring traffic participants from ground truth annotations with only "car" as instances.
Step 2 → acquiring frame-based instance information:
if POR $> 0.2\%$ **then**
⇒ start tracking instance;
⇒ collecting 3D tracking boundingbox: $[x, y, w, h, d_{min}, d_{max}]$;
⇒ collecting semantic labeling image I_{seg};
⇒ collecting 2D car pose from RGB input I_{car};
else if POR $< 0.1\%$ **then**
⇒ stop tracking.
end if
Step 3 → splitting sequences into tracklets of 23 frames with stepsize 1:
⇒ first 15 frames (5Hz, 3 sec) as training input;
⇒ next 8 frames (1.6 sec) as the held-up future trajectory to be predicted.

The focus of this paper is on car trajectory prediction on highways which corresponds to sequence number 1 in the SYNTHIA dataset. We are especially concerned about the cars that are relatively close to the driver. In order to decide quantitatively which vehicles are close to the host car, we define pixel occupancy ratio *(POR)* as the ratio between the total number of pixels of the tracking vehicle and the total number of pixels of the camera input. As in accordance with [6], we set the reliable car perception as 30 meters so as to guarantee satisfactory control quality when the speed of the host car does not exceed 72km/h. Since depth information is also provided alongside in the SYNTHIA dataset, we back-project the segmented instance region onto the depth maps and estimate that the POR of 0.2% and 0.1% correspond to roughly 30 meters and 100 meters of relative distance between the host car and the tracked vehicle. Hence, we start tracking the target when the POR of the detected vehicle is larger than 0.2% so as to suffice the safety distance for driver to make timely reaction and stop tracking the vehicle when the POR is smaller than 0.1% indicating the target car is too far to influence driving policy. We follow the prediction time allocation as in [48] that the historical time-span is 3 seconds (corresponds to 15 frames with 5 Hz frame rate in the SYNTHIA dataset). However, in contrast with [48] where only the next 1/3rd of a second is predicted, we strive to predict the next 1.6 second's future trajectory as mentioned previously that it's the reaction time accident reconstruction specialists commonly used [34].

We follow the traditional paradigm for visual tracking to collect target's bounding box: $[x, y, w, h]$ of each frame. Moreover, the minimum relative distance d_{min} and maximum relative distance d_{max} can also be acquired from the depth camera. The train/valid/test are split as 80%/10%/10% of the total tracklets and the total numbers

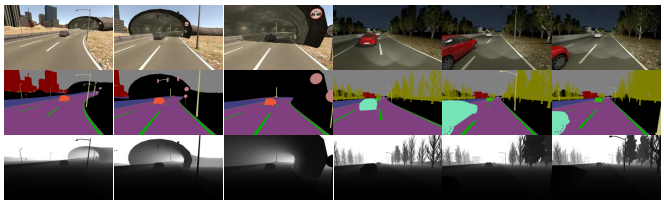


Figure 5: Examples of collected dataset for vehicle trajectory prediction. From top to bottom: RGB, semantic labeling, depth maps for three continuous frames under two dynamic light conditions.

	#train	#valid	#test
tracklets	10400	1300	1280
car detection	7832	976	986
semantic segmentation	7841	977	987

Table 1: Dataset statistics

are shown in Tab. 1. Some sample frames with semantic labels and depth information is shown in Fig. 5.

4.2. Implementation Details

4.2.1 Traffic participants detection

To reliably detect traffic participants, we compared two state-of-the-art approaches: SSD [32] and Faster-RCNN [39]. We use the pretrained network on the PASCAL VOC detection dataset [16] which has 20 classes and fine-tune the network on the SYNTHIA with two classes: cars *v.s.* background. A comprehensive survey of trade-offs for modern convolutional object detectors is presented in [29] and we refer keen readers to the aforementioned paper for a more complete comparison to achieve the right speed/memeory/accuracy balance for a given application and platform.

As is also pointed out in the paper [29] that SSD, albeit is less advantageous in detecting small objects, it's very competitive for detecting larger objects. In this paper, we are mostly concerned about cars that are close to the driver. Hence, given 0.1% POR as the cut-off threshold, the presented problem favors detectors that are robust in detecting larger objects. Tab. 5 verifies that SSD is indeed more competitive when objects of interests are large in our problem set. We also compare the influence of the cut-off threshold for various confident scores and non-maximum suppression (NMS) thresholds. It can been seen that both meta-architectures are robust to cut-off confident scores. Allowing larger NMS threshold enables the detector to have slightly higher recall, it comes at a cost of multiple redundant overlapping detections and much lower f-score overall. In the following experiment, we adopt SSD with a confident score of 0.5 as the cut-off threshold and 0.45 as the default jaccard overlap for NMS.

4.2.2 Instance association via tracking

To associate traffic participants across different frames (*i.e.*, instance association), we creatively combine detecting with tracking. For this task, we adopt the real-time scale adaptive tracker fDSST [11] that achieves 50 FPS with scale estimation. For every newly detected target whose POR is larger than the predefined threshold of 0.2%, we initiate the tracker with a unique target ID.

The tracker's target model is updated every time when there is a detection whose jaccard overlap is larger than 0.3. We argue that force update of tracker model via the detection template is essential when the target undergo rapid out of plane rotations. Trackers [37, 13, 45, 10] focus on the "class-agnostic" settings. In the context of autonomous driving, however, we know in advance the classes of object of interests for tracking. Therefore, with this extra source of information, we propose to use the neural nets object detection result as a more robust template to update the tackers model to overcome the problem of model drifting. Hence the tracker is served solely as instance associator for the traffic participant. In return, the instance association scheme helps to alleviate the problem of false positive detection (*c.f.* Fig. 8) from the detection result. The target's tracker will terminate under either of the following two conditions: if the target's POR is smaller than 0.1% (*i.e.*, the vehicle is too far way from the host car); or if there are more than 5 frames of detection absence (*i.e.*, the target could be one false positive detection from the vehicle detector).

4.2.3 Trajectory prediction

For the task of sequential prediction, both FtF and FtS share the same base network structure: a 3-layer recurrent net with 100, 300, 300 LSTM cells. For SEG-LSTM, since the focus of the scene context is high-way driving, auxiliary information is the scene semantics with number of channels $\mathcal{C}$ sets as 2 ("car" and "road"). We adopt the state-of-the-art semantic segmentation networks: Dilated Residual Networks (DRN) [49] and retrain on the SYNTHIA corpus. DRN achieves 68% and 82% pixel mAP on the car and road categories. The input semantics $\mathcal{A}$ are one-hot encoded in the dimension of $95 \times 160 \times 2$. The CNN for learning high level semantics is composed of 4 convolutional blocks. Each block includes a convolutional layer, a ReLU activation layer, a maxpooling layer and a dropout layer (as we have observed overfitting during network training). Each convolutional layer has four channels with kernel of size 3.

For the social tensor configuration, we incorporate all vehicles in the same time-step as their neighbors. In contrast with [2] where a spatial constraint is required in order to be counted as a neighbor, all the vehicles in the same scene are relevant for the social interactions under our high-way driving context. Grid size is set as 4 and the embedding size is 64 so the social tensor $\mathcal{S}$ is a $4 \times 4 \times 64$ matrix. The learning is set to 0.001 and RMS-prop [43] with a gradient clipping of 3 to avoid divergence.

Configuration 2D	mean coverage (%)	center error (pixel)
Kalman Filter	54.31	24.84
FtF	66.47	23.43
FtS	73.93	15.32
SEG-LSTM	**77.53**	**12.63**

Table 2: Results of 2D performance evaluation (pixel as unit) on mean coverage (higher is better) and center error (lower is better).

Configuration 3D	mean coverage (%)	center error (meter)
FtF	35.67	0.7122
FtS	46.69	0.3686
SEG-LSTM	**48.49**	**0.3298**

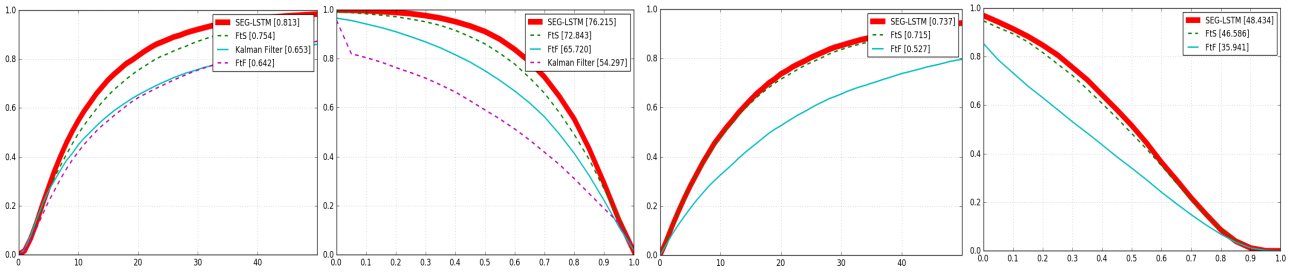
Table 3: Results of 3D performance evaluation (centimeter as unit) on mean coverage (higher is better) and center error (lower is better).

4.3. Evaluation

Baseline: Kalman filter, also known as linear quadratic estimation, is a popular technique for estimating the state of a system [42]. For the task of trajectory prediction, Kalman filters estimates a continuous state and gives a uni-modal distribution. The Kalman filter represents all distributions of the Gaussians and iterates two states: (1) measurement updates; (2) motion updates. In the baseline, we set the initial uncertainly covariance as 1e4, measurement noise and motion noise are all set with unit value.

2D Evaluation Methodology: Following the evaluation strategy of [46], all trajectories are compared using two measures: precision and success. Precision is measured as the distance between the centers of the ground truth bounding box and the corresponding tracker generated bounding box. The precision plot shows the percentage of tracker bounding boxed within a given threshold distance in pixels of the ground truth. To rank the prediction performance, the conventional threshold of 20 pixels (*P20*) is adopted. Success is measured as the intersection over union of pixels. The success plot shows the percentage of tracker bounding boxes whose overlap score is larger than a given threshold and the trackers are ranked according to the Area Under Curve (*AUC*) criteria.

3D Evaluation Methodology: Given depth map and the parameter of camera focal lens, we can formalize the problem into a 3D occupancy grid estimation. The real world coordinates $[x_r, y_r, w_r, h_r]$ can be obtained according to



(a) 2D Precision plot (b) 2D success plot (c) 3D precision plot (d) 3D Success plot

Figure 6: 2D and 3D space precision plot and success plot. x-axis for (a) to (d) are: pixel, IOU, centimeter and IOU.

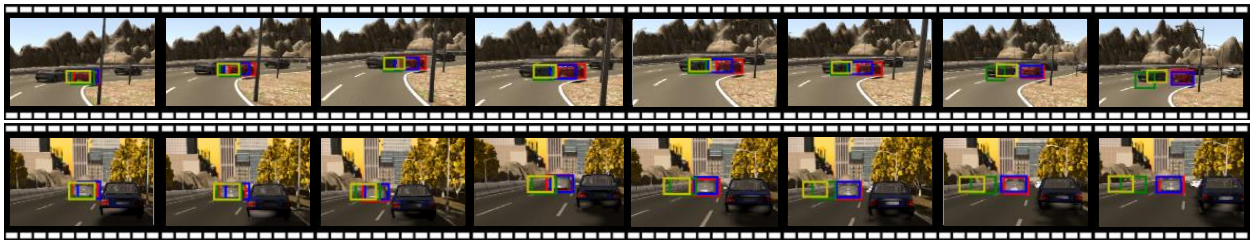


Figure 7: The prediction results in SYNTHIA dataset. Red: ground truth; Blue: SEG-LSTM; Green: FtS; Yellow: FtF. The results using SEG-LSTM network show that by utilizing the scene semantics (*e.g.*, road, cars), it's able to direct the predicted trajectory from the recurrent net within the most plausible path. Top: SEG-LSTM can help detecting neighboring vehicles and restricts the viable path outside the space occupied by other vehicles. Bottom: SEG-LSTM confines the possible car trajectories along the space above road.

	Ave. disp. error	Final disp. error
Vanilla LSTM	0.0257	0.0440
Social LSTM	**0.0247**	**0.0418**

Table 4: Results comparing vanilla LSTM and social LSTM in unit space. *Ave. disp. error* is the mean square error over all estimated points of a trajectory and the true points. *Final disp. error* is the distance beween the predicted final destination and the true final destination at the end of the prediction period T_F.

Eq. 3 4. The 3D jaccard index can be similarly formalized as the volume intersection over union: $\frac{V_g \cap V_p}{V_g \cup V_p}$, where V_g, V_p are the ground truth and predicted space occupancy. 3D center error correspondingly is defined as the mean square root in 3D space.

Learning with auxiliary information: The 2D and 3D evaluation results in Fig. 6 and Tab. 2 show that our proposed system consistently outperforms the baseline Kalman filter. Kalman filter is intrinsically a linear model and it's difficult to handle highly nonlinear driving trajectories in a relative coordinate. The auxiliary learning with semantic information in SEG-LSTM also consistently outperforms other two temporal models. Fig. 7 shows the advantage of using SEG-LSTM: by utilizing the scene semantics (*e.g.*, road, cars), it's able to direct the recurrent net to output the most plausible path for vehicles and avoid collision with neighboring vehicles.

Learning with social tensor: On average, there is around 3 cars of interests in a high-way driving scene. Tab. 4 shows that considering social interactions between cars improves the prediction accuracy than the isolation case. And we believe that in a more complex social scene such as urban driving, the improvement will be more pronounced.

5. Conclusions

In this paper, mimicking the human driver reaction time, we advocate to solve the problem of predicting traffic participants future trajectory given historical information. We propose a two-staged system incorporating various visual inputs with a temporally recurrent neural network. The recurrent network takes semantic information as an auxiliary input and further improve the predicting accuracy with the incorporation of social factor. Future works include incorporating vehicle pose from the RGB/depth input as extra source of "auxiliary information" and conducting experiments on the real-world images.

References

[1] Udacity. public driving dataset. `https://www.udacity.com/self-driving-car`, 2017.

[2] 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*, 2016.

[3] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[5] R. Chalasani and J. C. Principe. Deep predictive coding networks. *arXiv preprint arXiv:1301.3541*, 2013.

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] A. Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 2013.

[9] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[11] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[12] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*. Springer, 2016.

[13] M. Danelljan, K. F. S. Robinson, Andreas, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. *European Conference on Computer Vision*, 2016.

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[15] T. Egner, J. M. Monti, and C. Summerfield. Expectation and surprise determine neural population responses in the ventral visual stream. *Journal of Neuroscience*, 2010.

[16] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 2015.

[17] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, 2016.

[18] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013.

[19] R. Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[20] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[21] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[22] R. A. Hadi, G. Sulong, and L. E. George. Vehicle detection and tracking techniques: a concise review. *Signal & Image Processing*, 2014.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[25] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*. Springer, 2016.

[26] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[27] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[28] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. *International Conference on Machine Learning*, 2015.

[29] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint:1611.10012*, 2017.

[30] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] J. Kim and J. Canny. Interpretable learning for self-driving cars by visualizing causal attention. *International Conference on Computer Vision*, 2017.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, 2016.

[33] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[34] D. V. McGehee, E. N. Mazzae, and G. S. Baldwin. Driver reaction time in crash avoidance research: Validation of a driving simulator study on a test track. In *Proceedings of the human factors and ergonomics society annual meeting*, 2000.

[35] M. Mueller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. UE4Sim: A Photo-Realistic Simulator for Computer Vision Applications. *ArXiv e-prints*, Aug. 2017.

[36] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In *European Conference on Computer Vision*, 2016.

[37] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, and J. L. M.-H. Yang. Hedged deep tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, 2016.

[38] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2016.

[39] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015.

[40] 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*, 2016.

[41] E. Santana and G. Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

[42] S. Thrun. Artificial intelligence for robotics course. *Udacity, lecure note*, 2016.

[43] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.

[44] A. Vemula, K. Muelling, and J. Oh. Social Attention: Modeling Attention in Human Crowds. *International Conference on Robotics and Automation*, 2018.

[45] D. Wu, W. Zou, X. Li, and Y. Zhao. Kernalised multi-resolution convnet for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshop*, 2017.

[46] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[47] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[48] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. *IEEE conference on computer vision and pattern recognition*, 2017.

[49] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Appendices

A. Traffic participants detection

B. Instance association via tracking

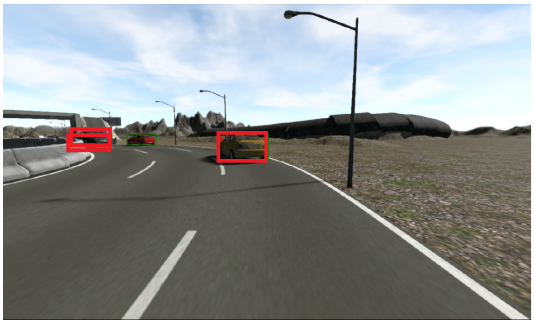


Figure 8: Instance association can help to disambiguate the false positive by enforcing temporal consistency requirement. Due to the high recall requirement for vehicle detection, there could be correspondingly higher false positive rate (*e.g.*, double detection of red boxed in the far left). Instance association across multiple frames will discard the spurious detection while remain temporal consistency.

C. 2D and 3D occupancy grid

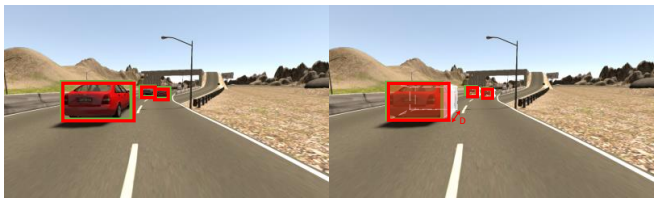


Figure 9: The evaluation of 2D and 3D prediction. 2D: The bounding box is set to cover the whole vehicle. 3D: The depth channel is combined with 2D bounding box to predict 3D occupancy.

conf		0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Faster-RCNN	precision	81.2	81.2	81.1	81.1	81.1	81.0	81.0	80.7	80.5	80.0
	recall	91.4	91.4	91.3	91.3	91.3	91.3	91.3	90.9	90.7	90.2
	f-score	86.0	86.0	86.0	86.0	86.0	86.0	86.0	85.5	85.3	84.8
SSD (NMS:0.60)	precision	82.1	81.9	81.8	81.8	81.6	81.1	80.9	80.4	80.0	79.2
	recall	**92.7**	92.5	92.3	92.3	92.0	91.5	91.3	90.7	90.2	89.4
	f-score	87.1	86.9	86.7	86.7	86.5	86.0	85.8	85.2	84.8	84.0
SSD (NMS:0.45)	precision	**92.0**	91.8	91.7	91.4	91.1	91.0	90.5	90.2	89.4	87.4
	recall	92.3	92.1	92.0	91.8	91.4	91.3	90.8	90.5	89.7	87.7
	f-score	**92.2**	92.0	91.8	91.6	91.3	91.1	91.1	90.4	89.6	87.6

Table 5: Comparison of SSD and Faster-RCNN for vehicle detection on the collected SYNTHIA dataset. It can be seen that: (1) both meta-architectures are robust to the cut-off confident thresholds. (2) SSD is more competitive in the collected dataset when targets of interest are large.