2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9341020

End-to-end Autonomous Driving Perception with Sequential Latent Representation Learning

Jianyu Chen, Zhuo Xu and Masayoshi Tomizuka

***Abstract*— Current autonomous driving systems are composed of a perception system and a decision system. Both of them are divided into multiple subsystems built up with lots of human heuristics. An end-to-end approach might clean up the system and avoid huge efforts of human engineering, as well as obtain better performance with increasing data and computation resources. Compared to the decision system, the perception system is more suitable to be designed in an end-to-end framework, since it does not require online driving exploration. In this paper, we propose a novel end-to-end approach for autonomous driving perception. A latent space is introduced to capture all relevant features useful for perception, which is learned through sequential latent representation learning. The learned end-to-end perception model is able to solve the detection, tracking, localization and mapping problems altogether with only minimum human engineering efforts and without storing any maps online. The proposed method is evaluated in a realistic urban driving simulator, with both camera image and lidar point cloud as sensor inputs. The codes and videos of this work are available at our github repo[†] and project website[‡].**

I. Introduction

Motivated by the potential social impact and fueled by the recent advances in both hardware (sensor technologies such as lidar) and software (artificial intelligence techniques such as deep learning), massive efforts from both industry and academia have been invested to autonomous driving during the last decade. Start from the DARPA urban challenges [22], [29], a number of autonomous vehicle system demonstrations have been performed. Automotive industry giants such as Benz [33], and IT giants such as Google [2] are competing to develop the first commercial fully autonomous vehicle.

A typical autonomous driving system is organized as a two parts architecture [1], [32]: a perception system, and a decision making system, as shown in Fig.1. The perception system is composed of multiple subsystems including detection [31], tracking [21], localization and mapping [4]. These systems, together with offline collected maps, transform the raw sensor inputs (e.g, camera RGB images and lidar point clouds) to useful information such as surrounding vehicles' poses, ego vehicle's pose, and the local semantic map centered around the ego vehicle. On the other hand, the decision making system is divided into subsystems including routing [3], behavior prediction [28], decision & planning [6], [9], and control [24] that work together to generate a control command (e.g, steering angle, throttle and braking) to drive the autonomous car.

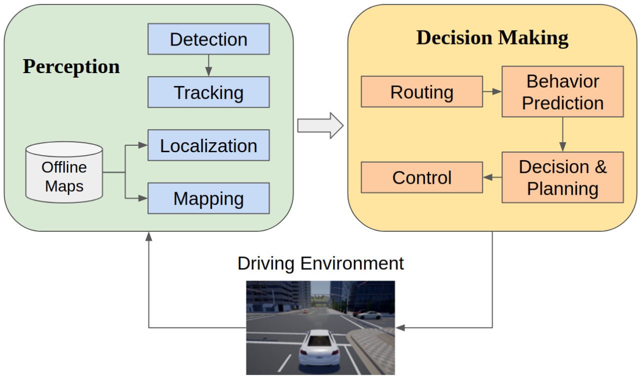


Fig. 1: Typical architecture of autonomous driving systems, which is composed of the perception part and decision making part with multiple subsystems.

Although this highly modularized architecture works well in a few driving tasks, it starts to touch its performance limitations because (1) too much human heuristics can lead to inappropriate perception results and driving behaviors; and (2) Too many complicated subsystems are making the whole system expensive to scale and maintain. Alternatively, end-to-end architectures might avoid those limitations, as the driving models can be learned and continuously optimized from data, without much hand-engineered involvement. However, although efforts have been made to build such autonomous driving architectures [5], [7], [8], an end-to-end system with decision making in the loop can only be demonstrated in simulations, real world applications are still infeasible due to safety issues. On the other hand, the perception system alone is suitable to be designed in an end-to-end form, since it does not require online driving exploration.

In this paper, we propose an end-to-end approach for the autonomous driving perception system defined in Fig.1. This method enables us to solve the detection, tracking, localization and mapping problems altogether, by learning a sequential latent representation model. The learned model is able to simultaneously provide accurate estimation of surrounding vehicle poses, ego vehicle global pose, and local semantic roadmap. With this end-to-end approach, we only need minimum human engineering efforts to obtain a fully functional perception system, and no maps are needed online. Furthermore, the fusion of these subsystems helps improve

J. Chen, Zhuo. Xu and M. Tomizuka are with Department of Mechanical Engineering, University of California, Berkeley, CA94720, USA. Email: {jianyuchen, zhuoxu, tomizuka}@berkeley.edu

[†] https://github.com/cjy1992/detect-loc-map

[‡] https://sites.google.com/berkeley.edu/e2e-percep

the performance of surrounding vehicle pose estimation.

The remainder of this paper is organized as follows. Section II summarizes existing works for autonomous driving perception. Section III analyzes the subsystems of a typical perception system to help us better understand their purposes and principles. Details of our proposed method is introduced in section IV. Section V shows the experiments and results, while section VI concludes the paper.

II. Related Work

It is crucial to perceive the environment and extract useful information. This mainly includes vehicle detection, tracking, localization and mapping.

A. Vehicle Detection

Vehicle detection refers to estimating the position, heading and size of surrounding vehicles. There are three main subclasses: (1) Image-based vehicle detection generates bounding boxes on front-view camera image [26], [27]; (2) Semantic segmentation assigns each pixel of the image with a class label. Pixels belonging to same objects have the same label [14], [20]; and (3) 3D object detection obtains the 3D poses (or bird-eye poses) of surrounding vehicles, which is usually achieved with the help of lidar point cloud [10], [30].

B. Vehicle Tracking

Direct prediction from the vehicle detection system is often insufficient, more accurate vehicle state needs to be estimated given historical detection. Typical vehicle tracking methods have two phases: (1) data association to connect objects between frames [15], [23]; and (2) filtering methods such as Kalman Filters and Particle Filters to smooth the vehicle dynamics [12], [25].

C. Localization and Mapping

Localization is the task of estimating ego vehicle pose relative to a reference frame in a map, which can be either a raw point cloud map or an annotated semantic map, depending on the algorithm we choose. There are two main approaches: (1) Simultaneous localization and mapping (SLAM) makes map online and localize the ego vehicle in the map at the same time [4]; and (2) A priori map-based localization that estimate the ego vehicle pose by finding the best match to a detailed a priori map [19].

III. Preliminary

In this section, we will analyze the typical subsystems in an autonomous driving perception system, including detection, tracking, localization and mapping. They are reformulated into graphical models. This helps us better understand their purposes and relationships to fuse them into a single end-to-end framework.

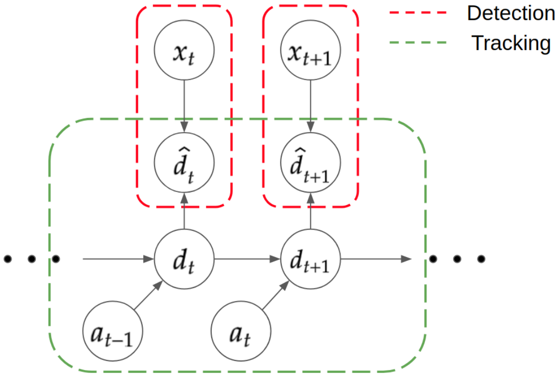


Fig. 2: Graphical model of typical detection and tracking systems. The red block represents the detection system. The green block represents the tracking system.

A. Typical Detection and Tracking Systems

The goal of the detection and tracking subsystems is to accurately estimate the states of surrounding vehicles. This includes their positions and heading angles relative to the ego vehicle, as well as their length and width. Current detection and tracking systems are divided into two subsystems. First, the detection subsystem predicts the vehicles' poses based on single frame sensor inputs. Then, the tracking subsystem smooths the results from the detection system based on its historical outputs.

These processes can be interpreted as a graphical model, as shown in Fig 2. The red block represents the detection subsystem, which contains a detection model $p\left(\hat{d}_t|x_t\right)$ that maps the sensor inputs x_t to an estimation of the surrounding vehicles' poses $\hat{d}_t$. This part is usually performed by fitting a deep neural network model using supervised learning techniques. The green block represents the tracking subsystem, which estimates the true vehicles' poses d_t given historical outputs of the detection system $\hat{d}_{1:t}$ and sometimes the historical ego vehicle actions $a_{1:t}$. This estimation problem can be formulated as a filter problem $p\left(d_t|\hat{d}_{1:t}, a_{1:t}\right)$ which is usually solved by Kalman filters with hard-coded transition model $p\left(d_{t+1}|d_t, a_t\right)$ and observation model $p\left(\hat{d}_t|d_t\right)$.

B. Typical Localization and Mapping Systems

The localization and mapping system needs to accurately estimate the pose of the ego vehicle in the coordinate of a global map. Then a local semantic map indicating road geometry, topology and traffics will be obtained, which are used for downstream planning and control tasks. The estimated global ego vehicle pose is also used for routing.

These processes can be interpreted as a graphical model, as shown in Fig 3. The global ego vehicle pose l_t is estimated based on historical sensor inputs and actions $p\left(l_t|x_{1:t}, a_{1:t}\right)$. The global map m can be edited offline or constructed online, depending on the methodology we use. After estimating the ego vehicle pose, a local semantic map S_t^{loc} is obtained by locating the ego vehicle on the global map, which requires

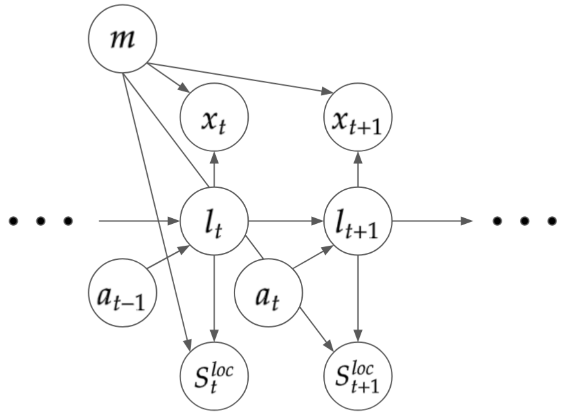


Fig. 3: Graphical model of typical localization and mapping system.

semantic annotations. Note that the global map m needs to be stored online.

IV. Proposed Method

In this paper, we propose a novel end-to-end autonomous driving perception system for simultaneous detection, tracking, localization and mapping. All the functionalities are fused in a single framework. A sequential latent representation learning process is performed to learn this end-to-end perception model.

A. End-to-end Autonomous Driving Perception

In general, the purpose of detection and tracking is to estimate surrounding vehicles' poses d_t, while the purpose of localization and mapping is to obtain the local semantic map S_t^{loc} and the global ego vehicle pose l_t. All these estimations are conditioned on historical sensor inputs $x_{1:t}$ and actions $a_{1:t}$. Therefore, the tasks of perception can be simplified by estimating the following conditional probability:

$$p\left(d_t, S_t^{loc}, l_t | x_{1:t}, a_{1:t}\right) \tag{1}$$

As stated in section III, typical methods estimate (1) by dividing it into multiple separate tasks, and tackle them one-by-one with lots of human engineering efforts. Different from typical methods, we propose to solve the problem of estimating (1) jointly. This is possible by assuming that there is a latent space summarizing all useful historical information. Then with this latent space, we can extract the information we need, such as surrounding vehicles' poses, road geometry, and ego vehicle pose. Inspired by works that learn latent representations with time sequence reasoning [5], [17], [18], we propose to formulate the end-to-end perception system as a single graphical model, as shown in Fig. 4. A more detailed architecture of our model in a single frame is shown in Fig. 5

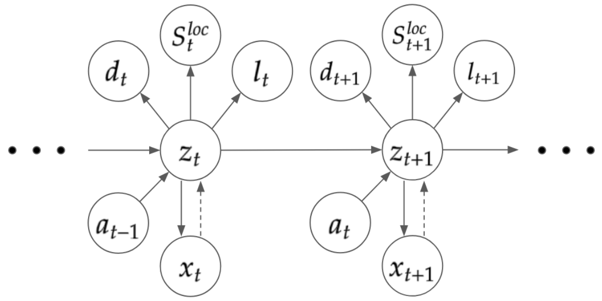


Fig. 4: Graphical model of end-to-end autonomous driving perception.

z_t is the latent variable we introduce, it represents a summary of historical information and is evolved with the latent dynamics $p\left(z_{t+1}|z_t, a_t\right)$. All relevant information about the environment are decoded from this latent, including the sensor inputs $p\left(x_t|z_t\right)$, surrounding vehicles' poses $p\left(d_t|z_t\right)$, local semantic roadmap $p\left(S_t^{loc}|z_t\right)$ and global ego vehicle pose $p\left(l_t|z_t\right)$. If we are able to estimate the distribution of the latent state given historical sensor inputs and actions $q\left(z_t|x_{1:t}, a_{1:t}\right)$, then (1) can be obtained by integrating out the latent state:

$$\begin{aligned} &p\left(d_t, S_t^{loc}, l_t | x_{1:t}, a_{1:t}\right) \\ &\qquad = \int p\left(d_t, S_t^{loc}, l_t | z_t\right) q\left(z_t | x_{1:t}, a_{1:t}\right) dz_t \end{aligned} \tag{2}$$

Generally we do not need to calculate the exact integration, but rather its expectation, which can be approximated by sampling:

$$\hat{z}_t \sim q\left(z_t|x_{1:t}, a_{1:t}\right) \quad \hat{d}_t, \hat{S}_t^{loc}, \hat{l}_t \sim p\left(d_t, S_t^{loc}, l_t|\hat{z}_t\right) \tag{3}$$

The next subsection will introduce how we can learn a model to estimate $q\left(z_t|x_{1:t}, a_{1:t}\right)$ and $p\left(d_t, S_t^{loc}, l_t|z_t\right)$ with sequential latent representation learning.

B. Sequential Latent Representation Learning

To learn appropriate models shown in Fig.4, we need to fit them with collected dataset. For convenience, we first denote a trajectory to be composed of sensor inputs, detection outputs, local semantic roadmaps, ego vehicle poses and actions:

$$\vec{x} = x_{1:t},\ \vec{d} = d_{1:t},\ S^{\vec{loc}} = S_{1:t}^{loc},\ \vec{l} = l_{1:t},\ \vec{a} = a_{1:t} \tag{4}$$

The dataset is then composed of this kind of trajectories collected while driving $\mathcal{D} = \left\{\left(\vec{x}^i, \vec{d}^i, \left(S^{\vec{loc}}\right)^i, \vec{l}^i, \vec{a}^i\right)\right\}_{i=1}^N$. The model can be fitted by maximizing the log likelihood of the data:

$$\begin{aligned} &\log \prod_{i=1}^N p\left(\vec{x}^i, \vec{d}^i, \left(S^{\vec{loc}}\right)^i, \vec{l}^i \mid \vec{a}^i\right) \\ &\qquad = \sum_{i=1}^N \log p\left(\vec{x}^i, \vec{d}^i, \left(S^{\vec{loc}}\right)^i, \vec{l}^i \mid \vec{a}^i\right) \end{aligned} \tag{5}$$

which can be maximized using stochastic gradient descent (SGD), which optimizes parametric functions with gradient descent. The gradient is estimated by sampling a batch of data points. To apply SGD to our problem, $p\left(\vec{x}, \vec{d}, S^{\vec{loc}}, \vec{l} \mid \vec{a}\right)$ needs to be composed of parametric functions, thus auto-differentiation tools such as TensorFlow can

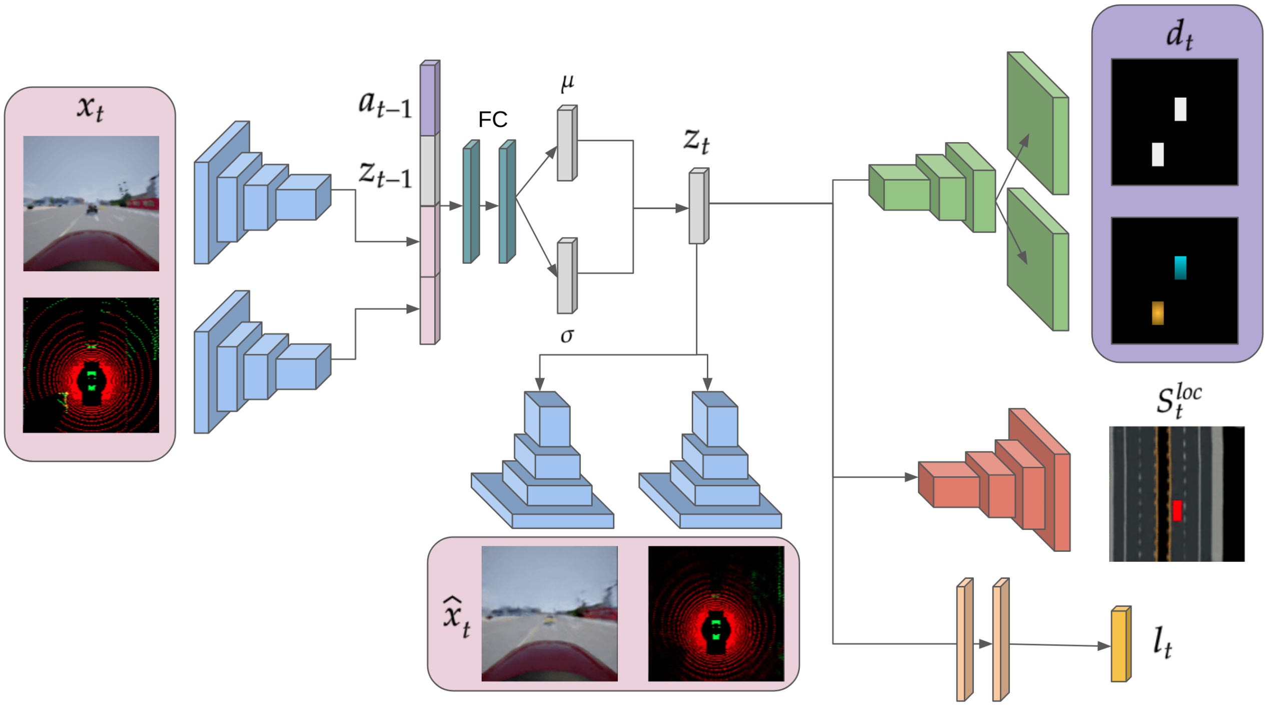


Fig. 5: Architecture of our proposed end-to-end perception model at a single frame.

be used to evaluate their gradients. Variational inference [16] can be applied to compute this log likelihood. First, introduce the latent variables $\vec{z} = z_{1:t}$:

$$\log p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l} \mid \vec{a}\right) = \log \int p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}, \vec{z} \mid \vec{a}\right) d\vec{z} \tag{6}$$

Then introduce a variational distribution $q\left(\vec{z}|\vec{x}, \vec{a}\right)$ into (6):

$$\begin{aligned}&\log p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l} \mid \vec{a}\right)\\ &\qquad = \log \int p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}, \vec{z} \mid \vec{a}\right) \frac{q\left(\vec{z}|\vec{x}, \vec{a}\right)}{q\left(\vec{z}|\vec{x}, \vec{a}\right)} d\vec{z}\end{aligned} \tag{7}$$

Now eliminate the integration in (7) by introducing expectation, and then apply Jensen's inequality:

$$\begin{aligned}\log p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}|\vec{a}\right) &= \log \mathop{\mathbb{E}}_{q(\vec{z}|\vec{x},\vec{a})}\left[\frac{p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}, \vec{z}|\vec{a}\right)}{q\left(\vec{z}|\vec{x}, \vec{a}\right)}\right]\\ &\geq \mathop{\mathbb{E}}_{q(\vec{z}|\vec{x},\vec{a})}\left[\log p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}, \vec{z}|\vec{a}\right)\right.\\ &\qquad\qquad \left. -\log q\left(\vec{z}|\vec{x}, \vec{a}\right)\right]\\ &= \text{ELBO}\end{aligned} \tag{8}$$

where ELBO stands for evidence lower bound. The original log likelihood can be maximized by maximizing this ELBO. Now derive $p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}, \vec{z}|\vec{a}\right)$ by probability factorization according to the PGM in Fig.4:

$$\begin{aligned}&p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}, \vec{z}|\vec{a}\right)\\ &\qquad = p\left(\vec{x}, \vec{d}, \vec{S^{loc}}, \vec{l}|\vec{z}, \vec{a}\right) p\left(\vec{z}|\vec{a}\right)\\ &\qquad = p\left(\vec{x}|\vec{z}\right) p\left(\vec{d}|\vec{z}\right) p\left(\vec{S^{loc}}|\vec{z}\right) p\left(\vec{l}|\vec{z}\right) p\left(\vec{z}|\vec{a}\right)\end{aligned} \tag{9}$$

And substitute (9) into (8), we have:

$$\begin{aligned}\text{ELBO} = \mathop{\mathbb{E}}_{q(\vec{z}|\vec{x},\vec{a})}&\left[\log p\left(\vec{x}|\vec{z}\right) + \log p\left(\vec{d}|\vec{z}\right) + \log p\left(\vec{S^{loc}}|\vec{z}\right)\right.\\ &\left. + \log p\left(\vec{l}|\vec{z}\right) + \log p\left(\vec{z}|\vec{a}\right) - \log q\left(\vec{z}|\vec{x}, \vec{a}\right)\right]\end{aligned} \tag{10}$$

Now derive the components in (10) by unfolding them with time. Considering the conditional dependence of PGM in Fig.4. The decoding models can be unfolded as:

$$\begin{aligned}\log p\left(\vec{x}|\vec{z}\right) &= \log \prod_{\tau=1}^{t} p\left(x_\tau|z_\tau\right) = \sum_{\tau=1}^{t} \log p\left(x_\tau|z_\tau\right)\\ \log p\left(\vec{d}|\vec{z}\right) &= \log \prod_{\tau=1}^{t} p\left(d_\tau|z_\tau\right) = \sum_{\tau=1}^{t} \log p\left(d_\tau|z_\tau\right)\\ \log p\left(\vec{S^{loc}}|\vec{z}\right) &= \log \prod_{\tau=1}^{t} p\left(S_\tau^{loc}|z_\tau\right) = \sum_{\tau=1}^{t} \log p\left(S_\tau^{loc}|z_\tau\right)\\ \log p\left(\vec{l}|\vec{z}\right) &= \log \prod_{\tau=1}^{t} p\left(l_\tau|z_\tau\right) = \sum_{\tau=1}^{t} \log p\left(l_\tau|z_\tau\right)\end{aligned} \tag{11}$$

The prior model can be unfolded using the latent state transition function:

$$\begin{aligned}\log p\left(\vec{z}|\vec{a}\right) &= \log\left[p\left(z_1\right)\prod_{\tau=1}^{t-1} p\left(z_{\tau+1}|z_\tau, a_\tau\right)\right] \\ &= \log p\left(z_1\right) + \sum_{\tau=1}^{t-1} \log p\left(z_{\tau+1}|z_\tau, a_\tau\right)\end{aligned} \tag{12}$$

The latent state inference model can be unfolded as:

$$\begin{aligned}\log q\left(\vec{z}|\vec{x},\vec{a}\right) &= \log\left[q\left(z_1|\vec{x},\vec{a}\right)\prod_{\tau=1}^{t-1} q\left(z_{\tau+1}|z_\tau,\vec{x},\vec{a}\right)\right] \\ &\approx \log\left[q\left(z_1|x_1\right)\prod_{\tau=1}^{t-1} q\left(z_{\tau+1}|z_\tau, x_{\tau+1}, a_\tau\right)\right] \\ &= \log q\left(z_1|x_1\right) + \sum_{\tau=1}^{t-1} \log q\left(z_{\tau+1}|z_\tau, x_{\tau+1}, a_\tau\right)\end{aligned} \tag{13}$$

Note here we approximate $q\left(\vec{z}|\vec{x},\vec{a}\right)$ and $q\left(z_{\tau+1}|z_\tau,\vec{x},\vec{a}\right)$ with $q\left(z_1|x_1\right)$ and $q\left(z_{\tau+1}|z_\tau, x_{\tau+1}, a_\tau\right)$ for simplicity. To obtain the exact values, bi-directional recurrent neural networks should be used to obtain the posterior probabilities conditioned on the whole trajectory sequence ($\vec{x}$, $\vec{a}$) [17].

Now we can unfold (10) with time:

$$\begin{aligned}\text{ELBO} \approx \mathop{\mathbb{E}}_{q(\vec{z}|\vec{x},\vec{a})} \Bigg[&\sum_{\tau=1}^{t} \log p\left(x_\tau|z_\tau\right) + \sum_{\tau=1}^{t} \log p\left(d_\tau|z_\tau\right) \\ &+ \sum_{\tau=1}^{t} \log p\left(S_\tau^{loc}|z_\tau\right) + \sum_{\tau=1}^{t} \log p\left(l_\tau|z_\tau\right) \\ &- \mathrm{D}_{\mathrm{KL}}\left(q\left(z_1|x_1\right)||p\left(z_1\right)\right) \\ &- \sum_{\tau=1}^{t} \mathrm{D}_{\mathrm{KL}}\left(q\left(z_{\tau+1}|z_\tau, x_{\tau+1}, a_\tau\right)||p\left(z_{\tau+1}|z_\tau, a_\tau\right)\right)\Bigg]\end{aligned} \tag{14}$$

and the ELBO is now decomposed to several simple networks, which will be illustrated in section IV-D.

C. Input Representations

1) **Sensor Input** x**:** We use two sensors to provide the observations, camera and lidar. For camera, the sensor input is a front-view RGB image, which can be represented by a tensor of $[0, 255]^{128\times128\times3}$, as shown in Fig.6(a). For lidar, we project the point clouds to the ground plane and render them into a 2D lidar image. The lidar image is represented by a tensor of $[0, 255]^{128\times128\times3}$, with each pixel rendered in red or green depending on whether there are lidar points at or above ground level existing in the corresponding pixel cell, as shown in Fig.6(b).

We use camera and lidar together because they are both important sensor sources and provide complementary information. Lidar point clouds provides accurate spatial information of other road participants and obstacles in 360 degrees of view. While the front-view camera is good at providing information of the road conditions.

2) **Detection Mask** d**:** The detection mask is composed of two branches: a classification branch outputing a 1-channel classification feature map and a regression branch outputing a 6-channel regression feature map. Both feature maps is in bird-eye view and their representation follows that in [31]. Briefly speaking, the classification feature map indicates the probability of each pixel belonging to a surrounding vehicle. While the regression feature map is composed of geometry information of the corresponding surrounding vehicle, e.g, x-y positions, heading, width and length of each detected vehicle (See Section 3.1.1 in [31] for details). With these two feature maps, we can use Non-Maximum Suppression (NMS) to get the final detection results. Thus d is composed of two tensors of $\mathbb{R}^{128\times128\times1}$ and $\mathbb{R}^{128\times128\times6}$ respectively.

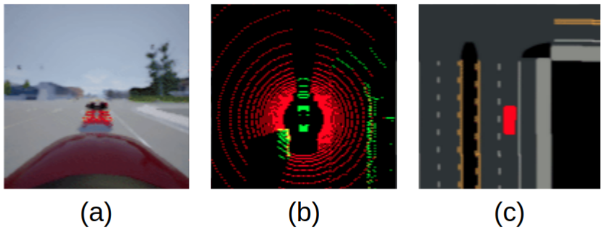


Fig. 6: Input representations. (a): Front camera RGB image; (b) lidar birdeye image; (c) local semantic roadmap. All images are with shape 128 × 128.

3) **Local Semantic Roadmap** S^{loc}**:** The local semantic roadmap is a map centered around the ego vehicle which includes semantic information about the road geometry, road topology and traffic rules. For example, the lane markings, drivable areas, and stop signs. It is an RGB image of $[0, 255]^{128\times128\times3}$, as shown in Fig.6(c).

4) **Global Ego Vehicle State** l**:** The global ego vehicle pose includes the information of ego vehicle's x-y position (in meter) and heading angle (in rad) in the global map's coordinate. It is a vector of $\mathbb{R}^3$.

D. Network Architectures

In this section, we will illustrate the detailed architectures of the networks in (14).

1) **Sequential Latent Model:** The sequential latent model includes the latent dynamics network $p\left(z_{\tau+1}|z_\tau, a_\tau\right)$, the filtering model network $q\left(z_{\tau+1}|z_\tau, x_{\tau+1}, a_\tau\right)$, $q\left(z_1|x_1\right)$, and the sensor inputs reconstruction network $p\left(x_\tau|z_\tau\right)$. Here we follow the two-layer hierarchical latent space structure as in [18], such that $z_\tau = [z_\tau^1, z_\tau^2] \in \mathbb{R}^{288}$ where $z_\tau^1 \in \mathbb{R}^{32}$ and $z_\tau^2 \in \mathbb{R}^{256}$. $p\left(z_{\tau+1}|z_\tau, a_\tau\right)$ consists of two fully connected layers with hidden units number 256, followed by a Gaussian output layer. $q\left(z_{\tau+1}|z_\tau, x_{\tau+1}, a_\tau\right)$ and $q\left(z_1|x_1\right)$ both consist of 5 convolutional layers ((32, 5, 2), (64, 3, 2), (128, 3, 2), (256, 3, 2), (256, 3, 2) and (256, 4, 1), with each tuple means (filters, kernel size, strides), as shown in Fig.7(a)) to first encode the sensor inputs x_t into features of size 256. Then two fully connected layers with hidden units number 256 are followed, with a Gaussian output layer. $p\left(x_\tau|z_\tau\right)$ both consist of 5 deconvolutional layers ((256, 4, 1), (256, 3, 2), (128, 3, 2), (64, 3, 2), (32, 3, 2), and (3, 5, 2), with each tuple means (filters, kernel size, strides), as shown in Fig.7(b)) with a fixed standard deviation of 0.1.

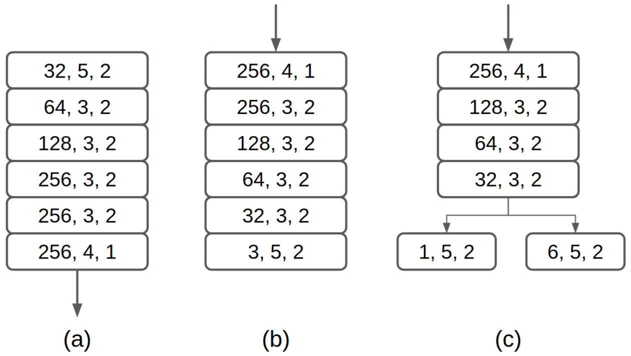
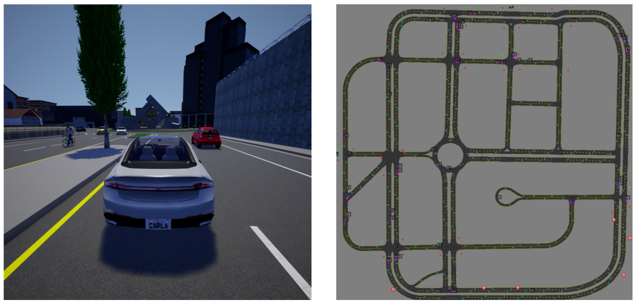


Fig. 7: Network architectures. (a): Image encoder for camera and lidar images; (b) Image decoder for camera, lidar and roadmap images; (c) Decoder for detection masks.



(a) Sample view of CARLA simulator

(b) Map layout of the simulated city

Fig. 8: Simulation environment

2) **Detection and Tracking Network:** The detection and tracking network is the network generating the detection mask $p\left(d_{\tau}|z_{\tau}\right)$. We deployed network architecture that is similar to the network architecture in [31]. The mask, including a one-channel tensor and a six-channel, is decoded from the latent by 4 deconvolutional layers ((256, 4, 1), (128, 3, 2), (64, 3, 2), (32, 3, 2), with each tuple means (filters, kernel size, strides)), and then a deconvolutional layer of (1, 5, 2) and (6, 5, 2) respectively, as shown in Fig.7(c).

3) **Localization and Mapping Network:** The localization and mapping network includes the local semantic roadmap decoder $p\left(S_{\tau}^{loc}|z_{\tau}\right)$ and the global ego vehicle state decoder $p\left(l_{\tau}|z_{\tau}\right)$. $p\left(S_{\tau}^{loc}|z_{\tau}\right)$ has the same architecture with $p\left(x_{\tau}|z_{\tau}\right)$, as shown in Fig.7(b). $p\left(l_{\tau}|z_{\tau}\right)$ is a two-layer fully connected neural network with hidden units number 256.

V. Experiments

A. Simulation Setup and Data Collection

We train and evaluate our proposed method on CARLA [11]. CARLA simulator is a high-definition open-source simulation platform designed for autonomous driving research. It simulates not only the driving environment and vehicle dynamics, but also the raw sensor data inputs such as camera RGB images and lidar point clouds. Fig.8 (a) shows a sample view of the driving simulation environment we use.

To collect the data for training, we navigate the ego vehicle in virtual town of CARLA. Fig.8 (b) shows the map layout of the town. It includes various urban scenarios such as intersections and roundabouts. The range of the map is $400m \times 400m$, with about $6km$ total length of roads. 100 vehicles are running autonomously in the virtual town to simulate a crowded urban environment. Both the ego vehicle and the surrounding vehicles will randomly choose a direction at intersections, then follow the route, while slowing down for front vehicles and stopping when the front traffic light becomes red. We run the ego vehicle for 50k environment steps and store the observations of each step into the training dataset.

B. Training Details

The model is trained with a batch size of 32 and learning rate 0.0001. The length of sequential model used for training is $t = 10$. The total iteration of training is 100k. We train three variants of our methods:

1) **Inputs and roadmap:** Both the raw sensor inputs and the local semantic roadmap are reconstructed when training. So the model is enforced to capture the features of both sensor inputs and the local semantic roadmap.

2) **No inputs reconstruction:** No raw sensor inputs are reconstructed. This is reasonable since we do not necessary need to output the reconstructed raw sensor inputs.

3) **No roadmap reconstruction:** No local semantic roadmaps are reconstructed. Then the model is not enforced to capture the features of the road geometry and traffic rules.

C. Evaluation Results

To evaluate our method, instead of evaluating on a collected test dataset, we directly put the ego vehicle in a random start point in the virtual town with surrounding vehicles, as described in V-A. During the navigation, the ego vehicle is collecting sensor inputs, encoding it to latent states, and decoding to perception outputs. We let the vehicle runs for 15k environment steps and evaluate the performance during this period of navigation.

Fig.9 shows examples of the perception output. The first row represents the original camera and lidar inputs, as well as the ground truth local semantic roadmap and surrounding vehicle bounding boxes. The second row shows the perception output given only the historical camera and lidar inputs. We can see that the model is able to generate accurate semantic roadmap and surrounding vehicle bounding boxes, even for the occluded area.

We also evaluate the statistic performance of the system according to typical evaluation metrics. For surrounding vehicle bounding box prediction, we plot the Precision-Recall Curve (PRC) and then compute the Average Precision (AP) as Area Under Precision-Recall Curve (AUC) [13]. The PRC and APs are computed under Intersection-Over-Union (IoU) of 0.1, 0.3, 0.5, and 0.7. Fig.10 shows the PRC of the methods. The APs are summarized in Table.I. we can see the variant that does not reconstruct the semantic roadmap has significantly worse performance than the other two variants.

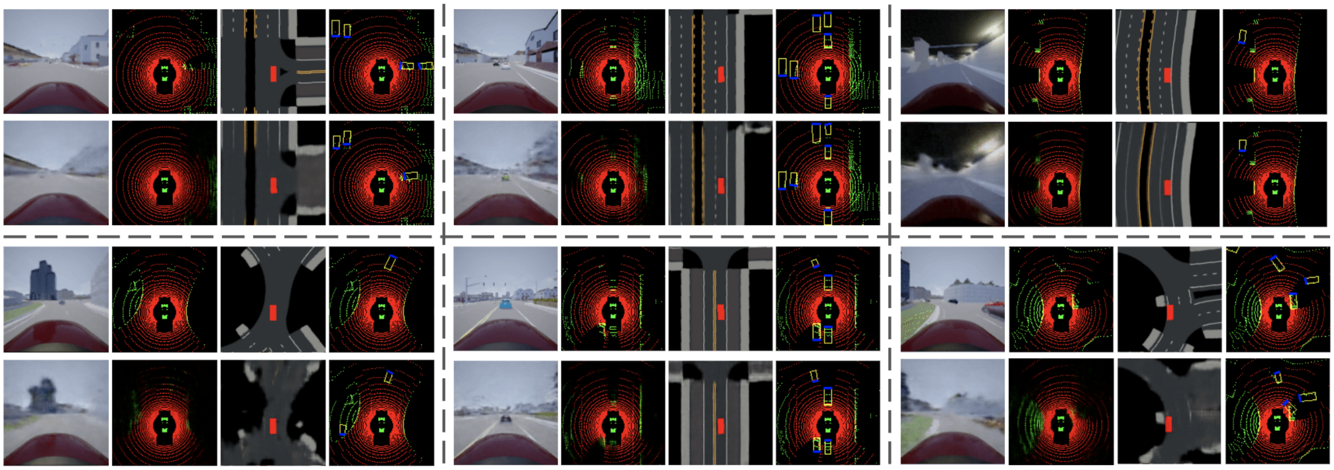


Fig. 9: Example results. The first row shows sensor inputs and ground truth, second row shows reconstructed outputs. Left to right: camera image, lidar image, local semantic roadmap, surrounding vehicle bounding boxes. Note that only camera and lidar inputs are given for the reconstruction, the ground truth semantic roadmap and bounding boxes are displayed here only for comparison.

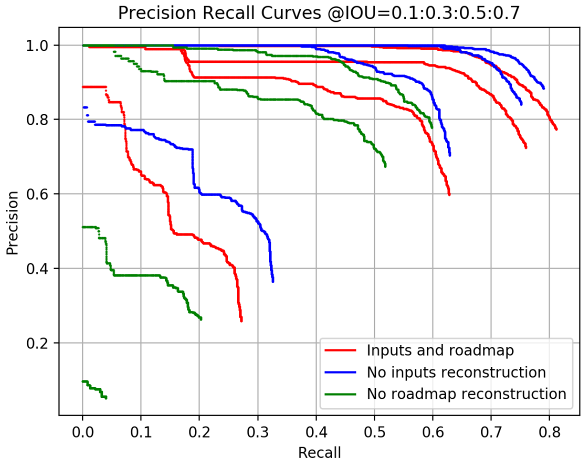


Fig. 10: Precision-Recall Curves for surrounding vehicles bounding boxes prediction.

TABLE I: Average precision for surrounding vehicles bounding boxes prediction.

	$AP_{0.1}$	$AP_{0.3}$	$AP_{0.5}$	$AP_{0.7}$
Inputs and roadmap	**79.4%**	72.0%	56.5%	16.8%
No inputs reconstruction	78.4	**74.4%**	**61.0%**	**22.1%**
No roadmap reconstruction	57.4%	45.3%	7.8%	0.3 %

This shows that fusing the information of map might improve the performance of detection.

For ego vehicle global state, we calculate its average prediction error. There are two values we care about, the location error (in meter) and heading error (in rad). Table.II shows the evaluation results. We can see that we get an average global location error of 7.1 meters, and heading error of 0.17 rad. This is obtained purely from the raw camera and lidar sensor inputs, with no GPS or stored global map used.

TABLE II: Average error of ego vehicle global pose estimation.

	Location (m)	Heading (rad)
Inputs and roadmap	11.4	0.33
No inputs reconstruction	8.6	**0.17**
No roadmap reconstruction	**7.1**	0.42

VI. Conclusion and Future Works

In this paper, we proposed and implemented an end-to-end autonomous driving perception system based on sequential latent representation learning. The system was able to replace the functionalities of the typical detection, tracking, localization and mapping subsytems with minimum human engineering efforts and without online stored maps. The method was evaluated in a realistic autonomous driving simulator, taking camera RGB image and lidar point cloud as sensor inputs. Evaluation results showed the learned model can obtain accurate surrounding vehicles' poses, local semantic roadmaps, and global ego vehicle pose based purely on historical raw sensor inputs.

Although the learned model performs reasonably well, it has a large space for improvement. The neural network architectures used in this paper are only the very basic ones, such as shallow convolutional layers. The size of input images is only 128 × 128, making it very hard to detect objects that are small or far away. In the future, we will deploy more advanced network architectures, and train the model on images with higher definition. We will also test this system with a downstream autonomous driving decision making system.

References

[1] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz, et al. Self-driving cars: A survey. *arXiv preprint arXiv:1901.04407*, 2019.

[2] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[3] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *Journal of Experimental Algorithmics (JEA)*, 15:2–1, 2010.

[4] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[5] J. Chen, S. E. Li, and M. Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *arXiv preprint arXiv:2001.08726*, 2020.

[6] J. Chen, C. Tang, L. Xin, S. E. Li, and M. Tomizuka. Continuous decision making for on-road autonomous driving under uncertain and interactive environments. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1651–1658. IEEE, 2018.

[7] J. Chen, B. Yuan, and M. Tomizuka. Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. *arXiv preprint arXiv:1903.00640*, 2019.

[8] J. Chen, B. Yuan, and M. Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2765–2771. IEEE, 2019.

[9] J. Chen, W. Zhan, and M. Tomizuka. Autonomous driving motion planning with constrained iterative lqr. *IEEE Transactions on Intelligent Vehicles*, 4(2):244–254, 2019.

[10] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[12] A. Ess, K. Schindler, B. Leibe, and L. Van Gool. Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29(14):1707–1725, 2010.

[13] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[15] S. Hwang, N. Kim, Y. Choi, S. Lee, and I. S. Kweon. Fast multiple objects detection and tracking fusing color camera and 3d lidar for intelligent vehicles. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 234–239. IEEE, 2016.

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters.(2015). *arXiv preprint arXiv:1511.05121*, 2015.

[18] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[19] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: science and systems*, volume 4, page 1. Citeseer, 2007.

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

[22] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[23] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M.-M. Meinecke. Stereo-camera-based urban environment perception using occupancy grid and object tracking. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):154–165, 2011.

[24] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[25] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[27] 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*, pages 91–99, 2015.

[28] C. Tang, J. Chen, and M. Tomizuka. Adaptive probabilistic vehicle trajectory prediction through physically feasible bayesian recurrent neural network. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3846–3852. IEEE, 2019.

[29] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[30] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[31] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[32] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: common practices and emerging technologies. *arXiv preprint arXiv:1906.05113*, 2019.

[33] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, et al. Making bertha drive—an autonomous journey on a historic route. *IEEE Intelligent transportation systems magazine*, 6(2):8–20, 2014.