2018 17th IEEE International Conference on Machine Learning and Applications

Future Trajectory Prediction via RNN and Maximum Margin Inverse Reinforcement Learning

Dooseop Choi, Taeg-Hyun An, Kyounghwan Ahn, and Jeongdan Choi
SW Contents Research Laboratory
Electronics and Telecommunications Research Institute (ETRI)
Republic of Korea
{d1024.choi, tekkeni, mobileguru, jdchoi}@etri.re.kr

***Abstract*—In this paper, we propose a future trajectory prediction framework based on recurrent neural network (RNN) and maximum margin inverse reinforcement learning (IRL) for the task of predicting future trajectories of agents in dynamic scenes. Given the current position of a target agent and the corresponding static scene information, a RNN is trained to produce the next position, which is the closest to the true next position while maximizing the proposed reward function. The reward function is also trained at the same time to maximize the margin between the rewards from the true next position and its estimate. The reward function plays the role of a regularizer when training the parameters of the proposed network so the trained network is able to reason the next position of the agent much better. We evaluated our model on a public KITTI dataset. Experimental results show that the proposed method significantly improves the prediction accuracy compared to other baseline methods.**

***Index Terms*—trajectory prediction, recurrent neural network, inverse reinforcement learning**

I. Introduction

Predicting future trajectories of agents in a dynamic scene has long been a great interest in many research fields such as video surveillance, autonomous driving. The distant future trajectory prediction from the past trajectories of the agent is, however, not easy because the target point of the agent is unknown in general, and even if the target position is known in advance, the motion of the agent is strongly affected by its neighbors and the static scene context such as road structure, buildings, traffic signs.

Intensive studies have been made in the literature for the accurate prediction of the future trajectory. The earlier works are usually based on the mathematical modeling of the agent's motion and the hand-crafted features [1]- [9]. Recently, with the success in training deep neural network and the availability of huge amount of training samples, several approaches that utilize recurrent neural network (RNN) have been proposed [10]- [12], and they showed promising results on publicly available datasets.

As we mentioned above, the motion of an agent in a dynamic scene is strongly affected by its neighbors and the static scene context. Therefore, considering such information for the prediction can improve the prediction accuracy, and many efforts have been made to effectively use the information for the prediction. Kitani et al. [4] proposed using the semantically rendered scene image in order to predict plausible paths for a pedestrian in a target task space through inverse reinforcement learning (IRL). Walker et al. [5] proposed learning the reward function, which describes the interactions between an agent and its surrounding, from visual inputs for the prediction of the plausible paths. Robicquet et al. [6] proposed an interaction model between two agents inspired by Social Force model [7] and used it to predict the future trajectories of the pedestrians. Kooij et al. [8] made an attempt to predict the future action of a pedestrian crossing the curbside based on the head image of the pedestrian, the distance between the ego vehicle and the pedestrian, and the spatial layout by the distance of the pedestrian to the curbside. Kretzschmar et al. [9] tried to learn the behavior of the pedestrians and their interactions via maximum entropy IRL. Alahi et al. [10] proposed a novel pooling method, called social pooling, to effectively learn the interactions between neighboring pedestrians via RNN. Recently, Lee et al. [11] proposed an RNN based future trajectory prediction framework that considers both the interactions between multiple agents and the static scene information. Xue et al. [12] used three RNNs to capture the pattern of the pedestrian's trajectory, the interactions between multiple agents, and the static scene information, separately.

IRL also has been utilized for the future trajectory prediction since predicting the future position from the past can be modeled as Markov decision process (MDP). However, previous works [4] [5] [9] assume that the goal position of an agent is given in advance, which makes them hard to be applied to general trajectory prediction task [11]. Lee et al. [11] made an attempt to recover the unknown reward function while training several RNNs for the future trajectory prediction. The recovered reward function, however, was not used to improve the prediction accuracy but to rank the multiple candidate trajectories. In this paper, we introduce maximum margin IRL framework into RNN-based future trajectory prediction. A reward function as well as RNN are simultaneously trained through the proposed loss functions, where the reward function plays the role of a regularizer for the weights of the RNN so that the trained network is able to reason the next motion of the agent much better.

The main contributions of this paper can be summarized as follows: (1) maximum margin IRL is first introduced into RNN-based future trajectory prediction framework, (2) min-max sampling method is proposed to train the RNN and

978-1-5386-6805-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICMLA.2018.00026

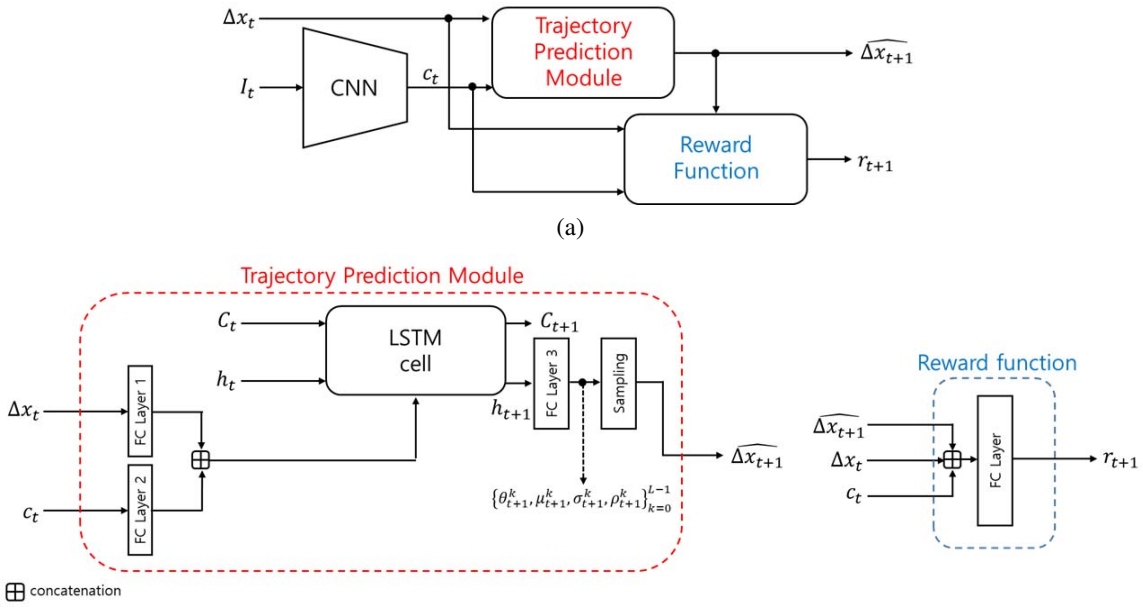

Fig. 1. (a) The overview of the proposed prediction framework. (b) The details of the proposed network.

the reward function simultaneously through the proposed loss functions in (3) and (4).

The rest of this paper is organized as follows. We briefly describe IRL in section II. The proposed network as well as its training method is explained in section III. The proposed method is evaluated in section IV and some conclusions are made in section V.

II. Review on Inverse Reinforcement Learning

The aim of reinforcement learning (RL) is to find the optimal policy for an agent, which achieves the maximum expected reward while the agent is moving from the initial state to the goal state [13]. However, the reward function is usually unknown in real-world problems. IRL was introduced to tackle this problem by recovering the reward function from the expert's demonstrations, and many variants have been proposed in the literature [14]- [19]. Maximum margin IRL is one of the variants and has been received a great attention owing to its performance and simplicity. Let us explain maximum margin IRL in details. A sequential decision making problem can be modeled by a Markov Decision Process (MDP) of tuple (S, A, T, γ, R), where S is a set of states; A is a set of actions; T is a set of state transition probabilities; γ is a discount factor; and R is the reward function we would like to recover. The reward function is usually modeled as $R = w^T\phi(s,a)$ where $w \in \mathrm{R}^k$ is a weight vector, $\phi : S \times A \rightarrow \mathrm{R}^k$ is a pre-defined mapping function, and s and a are samples from S and A, respectively. Given the expert's demonstrations $\{s_0^i, a_0^i, s_1^i, a_1^i, ...\}_{i=1}^m$, IRL tries to find w, which maximizes the expected reward of the policy drawn by the expert's demonstrations. Abbeel et al. [15] proposed an iterative algorithm that finds w and a policy jointly. The algorithm can be summarized as follows. Given an initial policy π^0, find w^0 that maximizes the difference between the expected reward from the expert, R_E and the expected reward from π^0, R_{π^0}. Next, find a new policy π^1 based on w^0 and add it in a policy set Π. Again, find w^1 that maximizes the difference between R_E and $R_{\pi^{max}}$, where $R_{\pi^{max}}$ is the maximum of the expected rewards available from the current Π. The above sequential steps are repeated until the pre-defined criterion is met. The algorithm does not necessarily recover the underlying reward function correctly, but finds a policy π^n whose performance is very close to the expert [15].

III. Proposed Framework

A. Problem Statement and Proposed Network

Let $\mathbf{x}_{i,t} = \{x_{i,t-N_{obs}-1}, x_{i,t-N_{obs}}, ..., x_{i,t}\}$ denote the past trajectory of the ith agent in a dynamic scene observed at time t, where $x_{i,t} \in \mathbb{R}^2$ represents the 2D coordinate of the agent at time t. Our goal is to predict the future trajectory $\mathbf{y}_{i,t} = \{y_{i,t+1}, y_{i,t+2}, ..., y_{i,t+N_{pred}}\}$ from $\mathbf{x}_{i,t}$ and the static scene information. Note that, in the rest of this paper, we omit index i for the simple expression. Figure 1 shows the overall architecture of the proposed network. We design our network to predict the next position offset $\Delta x_{t+1} = x_{t+1} - x_t$ (trajectory prediction module) and produce the corresponding reward r_{t+1} (reward function) given the current position offset $\Delta x_t = x_t - x_{t-1}$ and the corresponding static scene information $\mathcal{I}_t$ observed at time t. For the sake of easy explanation, we will call the *offset* the *position* in the rest of this paper.

a) Trajectory Prediction Module (TPM): Let c_t denote the output of the convolutional neural network (CNN) in

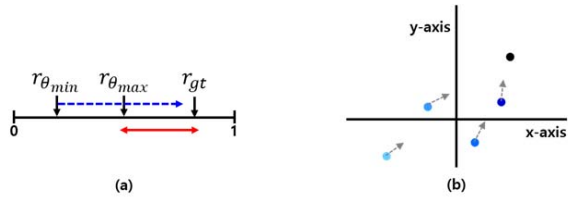

Fig. 2. Conceptual explanation of the proposed training method. (a) Reward values in the reward domain. (b) Positions in the pose domain.

Figure 1 when the static scene information $\mathcal{I}_t$ is input. Δx_t and c_t first pass through fully-connected (FC) layers, separately, and the outputs of the FC layers are concatenated to be used as input to our RNN with Long Short-Term Memory (LSTM) unit [16]. Another FC layer is finally used to produce the parameters of L Gaussian distributions, $\boldsymbol{\theta}_{t+1} = \{\theta^k_{t+1}, \mu^k_{t+1}, \sigma^k_{t+1}, \rho^k_{t+1}\}^{L-1}_{k=0}$, from the output of the RNN based on the assumption

$$\Delta x_{t+1} \sim \sum_{k=0}^{L-1} \theta^k_{t+1} \cdot \mathcal{N}(\mu^k_{t+1}, \sigma^k_{t+1}, \rho^k_{t+1}), \tag{1}$$

where $\mu^k_{t+1} = (\mu_x, \mu_y)^k_{t+1}$, $\sigma^k_{t+1} = (\sigma_x, \sigma_y)^k_{t+1}$, and ρ^k_{t+1}, respectively, are the mean, standard deviation, and correlation coefficient for the kth Gaussian. $\{\theta_k\}^{L-1}_{k=0}$ are the weight factors for the L Gaussian distributions and satisfy $\sum^{L-1}_{k=0} \theta_k = 1$.

b) Reward Function: The reward function R produces a reward value $r_{t+1} \in [0, 1]$ for choosing the next action $\Delta\hat{x}_{t+1}$ given the current state $\mathbf{s}_t = (\Delta x_t,\, c_t)$ as follows.

$$r_{t+1} = R(\Delta\hat{x}_{t+1}, \mathbf{s}_t) \tag{2}$$

We use a FC layer to calculate the reward value from $\Delta\hat{x}_{t+1}$ and $\mathbf{s}_t$. As seen in Figure 1, the CNN is used not only for TPM but also for the reward function. In addition, the parameters of the CNN is simultaneously trained via two loss functions, one for TPM and the other for the reward function as will be explained in the next section. Therefore, the proposed reward function is not a simple FC layer but is the CNN followed by the FC layer.

B. Proposed Losses and Training Method

We train our proposed network based on maximum margin IRL framework. The parameters of TPM (W_T) and the CNN (W_C) are learned by minimizing both the negative log-likelihood loss [20] and the negative log-reward loss given the fixed parameters of the reward function, W_R:

$$\mathcal{L}_T(W_C, W_T|W_R) = -\sum_{t=1}^{T}\Big(\underbrace{\mathbf{log}(P(\Delta x_t|\boldsymbol{\theta}_t))}_{log-likelihood\ loss} + \lambda\underbrace{\mathbf{log}(\beta^{(T-t)}R(\Delta x_{\boldsymbol{\theta_t}}, \mathbf{s}_{t-1}))}_{log-reward\ loss}\Big), \tag{3}$$

where $\Delta x_{\boldsymbol{\theta_t}}$ is a position sampled from $\mathcal{N}(\boldsymbol{\theta_t})$, λ is a weight factor trading off the likelihood loss against the reward loss,

Algorithm 1 TPM training via min-max sampling

Input: $\{\Delta x_t\}^T_{t=0}$, $\{\mathcal{I}_t\}^T_{t=0}$, W_T, W_R, W_C
Initialization: $\mathcal{L}_T = 0$, $t = 1$
while $t < T + 1$ **do**
  calculate $\boldsymbol{\theta}_t$ from Δx_{t-1} and $\mathcal{I}_{t-1}$ via TPM;
  sample Δx_{min} from $\{\mu^{min}_t, \sigma^{min}_t, \rho^{min}_t\}$;
  $\mathcal{L}_T \leftarrow \mathcal{L}_T - \Big(\mathbf{log}(P(\Delta x_t|\boldsymbol{\theta}_t)) + \lambda\mathbf{log}(\beta^{(T-t)}R(\Delta x_{min}, \mathbf{s}_{t-1}))\Big)$;
  $t \leftarrow t + 1$
end while
Update: $W_T \leftarrow W_T + \alpha\frac{\sigma\mathcal{L}_T}{\sigma W_T}$; $W_C \leftarrow W_C + \alpha\frac{\sigma\mathcal{L}_T}{\sigma W_C}$;

Algorithm 2 Reward function training via min-max sampling

Input: $\{\Delta x_t\}^T_{t=0}$, $\{\mathcal{I}_t\}^T_{t=0}$, W_T, W_R, W_C
Initialization: $\mathcal{L}_R = 0$, $t = 1$
while $t < T + 1$ **do**
  calculate $\boldsymbol{\theta}_t$ from Δx_{t-1} and $\mathcal{I}_{t-1}$ via TPM;
  sample Δx_{max} from $\{\mu^{max}_t, \sigma^{max}_t, \rho^{max}_t\}$;
  $\mathcal{L}_R \leftarrow \mathcal{L}_R - \mathbf{log}\Big(R(\Delta x_t, \mathbf{s}_{t-1}) - R(\Delta x_{max}, \mathbf{s}_{t-1}) + 1.0\Big)$;
  $t \leftarrow t + 1$
end while
Update: $W_R \leftarrow W_R + \alpha\frac{\sigma\mathcal{L}_R}{\sigma W_R}$; $W_C \leftarrow W_C + \alpha\frac{\sigma\mathcal{L}_R}{\sigma W_C}$;

β is a discount factor, and T is the length of the trajectory used for training. We let β be slightly smaller than 1.0 so that our TPM concentrates more on the distant future trajectory prediction.

The parameters of the reward function (W_R) and the CNN (W_C) are learned by minimizing the following maximum margin loss given W_T:

$$\mathcal{L}_R(W_C, W_R|W_T) = -\sum_{t=1}^{T}\mathbf{log}\Big(R(\Delta x_t, \mathbf{s}_{t-1}) - R(\Delta x_{\boldsymbol{\theta_t}}, \mathbf{s}_{t-1}) + 1.0\Big). \tag{4}$$

It can be found from (4) that the reward function is trained to maximize the margin between the rewards from the true position and the position estimated by TPM. On the other hand, TPM is trained to maximize the reward as well as the log-likelihood. The simultaneous minimization of the losses (3) and (4) can be regarded as a special case of the iterative algorithm presented by [15], where L Gaussians each plays the role of a policy in Π.

The remaining problem is how to sample a position from $\mathcal{N}(\boldsymbol{\theta_t})$ for the reward calculation during the simultaneous training. In this paper, we introduce our min-max sampling method. Let $\{\mu^{min}_t, \sigma^{min}_t, \rho^{min}_t\}$ and $\{\mu^{max}_t, \sigma^{max}_t, \rho^{max}_t\}$, respectively, denote the parameters of Gaussian with the smallest and the largest θ_t among $\{\theta^k_t\}^{L-1}_{k=0}$. Also, let $\Delta x_{\theta_{max}}$ and $\Delta x_{\theta_{min}}$ denote the positions sampled from $\{\mu^{max}_t, \sigma^{max}_t, \rho^{max}_t\}$ and $\{\mu^{min}_t, \sigma^{min}_t, \rho^{min}_t\}$, respectively. We let the reward function maximizes the reward margin

between Δx_t and $\Delta x_{\theta_{max}}$ through (4) while letting TPM maximizes the reward from $\Delta x_{\theta_{min}}$ through (3). We can explain the reason behind our method using Figure 2. In Figure 2-(a), r_{gt}, $r_{\theta_{max}}$, and $r_{\theta_{min}}$, respectively denote the reward from Δx_t, $\Delta x_{\theta_{max}}$, and $\Delta x_{\theta_{min}}$. In the middle of our simultaneous training, $r_{gt} \geq r_{\theta_{max}}$ and $r_{\theta_l} \geq r_{\theta_m}$ for $\theta_l > \theta_m$ may hold because the mixtured Gaussian distribution will be more dominated by the Gaussians with larger weights. As mentioned above, the reward function tries to maximize the margin between r_{gt} and $r_{\theta_{max}}$. (Red arrow in Figure 2-(a).) On the other hand, TPM tries to maximize $r_{\theta_{min}}$ so that it is closer to r_{gt}. (Blue arrow in Figure 2-(a).) Therefore, as the simultaneous training progresses, the positions sampled from each $\{\mu_t^k, \sigma_t^k, \rho_t^k\}$ will be closer to Δx_t as illustrated in Figure 2-(b). The black circle in the figure denotes Δx_t while the rests are the positions sampled from each $\{\mu_t^k, \sigma_t^k, \rho_t^k\}$. The circles with darker blue indicate the positions sampled from the Gaussians with larger θ_t^k. The proposed simultaneous training method via min-max sampling is summarized in Algorithm 1 and Algorithm 2. We run the two algorithms alternatively until the pre-defined criterion is met.

IV. Experiments

A. Dataset

To evaluate the propose method and other baselines, we utilized KITTI tracking dataset [21]. There are 21 videos in the dataset (from *0000* to *0020*). Each video is provided with Velodyne 3D laser scanning results, stereo camera images, GPS/IMU information of the ego vehicle, the relative distance between the ego vehicle and surrounding objects, and sensor calibration information. Each object in a video is provided with its class, the identification number, and 3D bounding box information so we could obtain the trajectories of the objects by using the center position of the 3D bounding box. There are total 938 unique objects in the dataset including the ego vehicle. We utilized 228 trajectories out of 938 trajectories since the excluded trajectories are too noisy or too short to be used as training samples. We, however, used the trajectories whenever we need the occupancy grid map [10] for our experiments. The total length of the selected sequences is 29049 frames. For training and validation, we made three dataset groups by randomly selecting 12 videos out of the 21 videos for each. This is because it took quite a long time to train over the 21 videos. Note that the prediction performances reported in the next section will be improved if the whole videos are used for the training. For each group, randomly selected 20 percent is used for validation and the rest is used for training.

The static scene information is obtained from Velodyne 3D scanning results. Point clouds from the laser scanning are first projected on the 2D world coordinate and accumulated for the whole time steps. Next, the accumulation result is manually segmented to produce the semantically rendered scene image. Figure 3 shows the scene image for video *0000*. The red, blue, and black pixels of the image correspond to drivable area, sidewalk, and the rests (building, grass etc), respectively. One pixel of the image corresponds to 0.33m in real world.

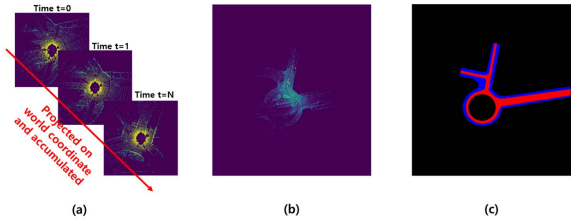

Fig. 3. (a) Velodyne point cloud accumulation step. (b) Accumulation result. (c) Semantically rendered scene image.

TABLE I
Details of Proposed Network

Layer	Size	Activation	Pooling
Convolutional Layer			
Conv.1	6 @ (3×3×3)	Relu	Max pooling (2, 2)
Conv.2	9 @ (3×3×6)	Relu	Max pooling (2, 2)
Conv.3	9 @ (3×3×9)	Relu	Max pooling (2, 2)
Trajectory Prediction Module			
FC. 1	(2×128)	Relu	-
FC. 2	(144×128)	Relu	-
FC. 3	(256×60)	Exp, Tanh	-
LSTM	(256×1)	-	-
Reward Function			
FC.	(148×1)	Sigmoid	-

B. Implementation Details

Table I summarizes the details of the proposed network. The static scene information for an agent at time t, $\mathcal{I}_t$, was obtained by cropping the semantic image around the position of the agent at time t, and we let the size of $\mathcal{I}_t$ be 48×48 pixels. We also let $\lambda = 0.0116$, $\beta = 0.99$, T=30 (3.0 sec), and L=10 (the number of Gaussians in (1)) during our simultaneous training, and they were chosen experimentally. The proposed network as well as other baselines were trained 300 epochs via Adam optimizer [22] with the initial learning rate 0.0001. All the trainable parameters were initialized by Gaussian random noise with standard deviation 0.01. To avoid exploding gradient in RNN, we applied gradient clipping with L2 norm of 10.0. We scaled all trajectories in the datasets by a factor of 0.1. We also applied random rotation to each trajectory and corresponding scenes in the training datasets for data augmentation. All models were implemented by using TensorFlow. Training took approximately one to five days per model for each group with a single GPU (NVIDIA GTX 1080ti).

C. Baselines and Evaluation Metrics

We compared the proposed model with three baselines:

- *LSTM* : TPM that predicts the next position only from the current position. It is trained via log-likelihood loss.
- *LSTM+O* : TPM that predicts the next position from the current position and the occupancy grid map [15]. It is trained via log-likelihood loss.
- *LSTM+I* : TPM that predicts the next position from the

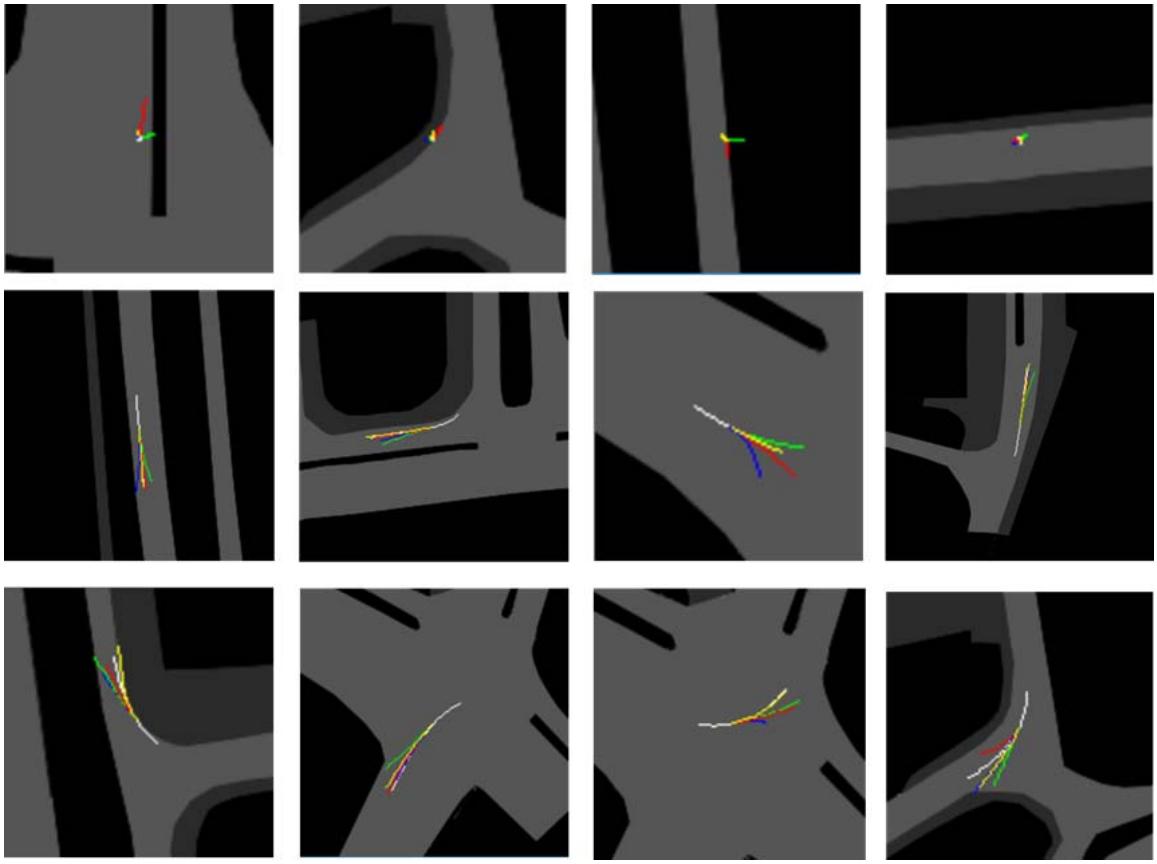

Fig. 4. Ground-truth trajectory (white) and its prediction results (blue, green, red, and yellow, respectively for LSTM, LSTM+O, LSTM+I, and IRL-LSTM+I) overlaid on the semantically rendered scene image.

TABLE II
AVERAGE DISPLACEMENT ERROR (METER)

prediction time	LSTM	LSTM+O	LSTM+I	IRL-LSTM+I
0.5 sec	0.39	0.35	0.37	**0.28**
1.0 sec	1.17	1.09	1.08	**0.81**
1.5 sec	2.17	2.07	1.94	**1.50**
2.0 sec	3.43	3.35	3.00	**2.36**

TABLE III
MISS RATE WITH 2M THRESHOLD

prediction time	LSTM	LSTM+O	LSTM+I	IRL-LSTM+I
0.5 sec	0.02	**0.00**	0.02	0.02
1.0 sec	0.13	0.13	0.09	**0.03**
1.5 sec	0.36	0.40	0.38	**0.22**
2.0 sec	0.65	0.68	0.56	**0.44**

current position and the static scene information. It is trained via log-likelihood loss.

• *IRL-LSTM+I* : TPM that predicts the next position from the current position and the static scene information. It is trained via Algorithm 1 and 2 (our method).

For evaluation, we let each algorithm observe 1 second (10 frames) and predict the next 2 seconds (20 frames) considering the application to autonomous driving. (Human drivers need the 1-2 seconds of reaction time for typical accident scenarios [23].) Displacement error (L2 distance between the prediction and ground-truth) and miss rate with 2m threshold are then calculated for the objective comparison. The threshold is determined based on that the length of common vehicles is about 4m.

D. Analysis

Table II and III, respectively, show the average displacement error and the miss rate at four prediction time steps. It is seen in the tables that IRL-LSTM+I (our method) shows the best performances among the four. LSTM+O shows the performances nearly the same as those of LSTM. This is because there are few agents interacting each other in KITTI dataset so the RNN could not obtain useful information from the occupancy grid map during the future trajectory prediction. However, the occupancy grid map will definitely help the RNN to predict the future trajectories of agents in dynamic scenes, where the agents actively interact each other. Therefore, one can consider incorporating the occupancy grid map into the proposed framework. LSTM+I shows the performances worse than IRL-LSTM+I. We can explain the reason using examples

shown in Figure 4. The figure shows the ground-truth trajectory (white) and its prediction results (blue, green, red, and yellow respectively for LSTM, LSTM+O, LSTM+I, and IRL-LSTM+I) overlaid on the semantically rendered scene image. Note that, for the visualization purpose, the scene images are shown in gray. As shown in the figure, LSTM+I predicts the trajectories relatively well when the agents move along the road. This is because the RNN effectively utilizes the road structure when predicting the future locations. However, when the agents keep staying at the initial position, LSTM+I fails to predict the future locations. (See the first row of Figure 4. LSTM+I produces false trajectories along the road structure.) This implies that a naive utilization of the static scene information can produce undesirable prediction results. In contrast, our method shows good prediction results on various situations.

V. Conclusion

In this paper, we presented a future trajectory prediction framework based on RNN and maximum margin IRL. We trained the RNN and the reward function simultaneously by minimizing the proposed loss functions via the min-max sampling method. During the simultaneous training, the reward function plays the role of a regularizer for the parameters of the proposed network even if the recovered reward function may not be the same as the true reward function. Experimental results prove that, through the simultaneous training, the proposed network is able to utilize the static scene information much better for the trajectory prediction.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) [No. 2017-0-00068, A Development of Driving Decision Engine for Autonomous Driving(4th) using Driving Experience Information].

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, 82(1):35-45, 1960.

[2] P. McCullagh and J. A. Nelder, "Generalized linear models," *Journal of basic Engineering*, volume 37, CRC press, 1989.

[3] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):283-298, 2008.

[4] K. M. Kitani, B. D. Zeibart, J. A. Bangnell, and M. Hebert, "Activity forecasting," In *European Conference on Computer Vision (ECCV)*, pages 201-214, Springer, 2012.

[5] J. Walker, A. Gupta, and M. Hebert, "Patch to future: Unsupervised visual prediction," In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVRP)*, pages 3302-3309, IEEE, 2014.

[6] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," In *European Conference on Computer Vision (ECCV)*, pages 549-565, Springer, 2016.

[7] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, 51(5):4282, 1995.

[8] J. F. P. Kooij, N. Schneider, F. Flohr, and D. M. Gavrila, "Context-based pedestrian path prediction," In *European Conference on Computer Vision (ECCV)*, pages 618-633, Springer, 2014.

[9] H. Kretzschmar, M. Kuderer, and W. Burgard, "Learning to predict trajectories of cooperatively navigating agents," In *IEEE 2014 International Conference on Robotics and Automation (ICRA)*, pages 4015-4020, IEEE, 2014.

[10] A. Alahi, K. Goel, V. Ramanathan, A. Robicque, L. Fei-Fei, and S. Savarese, "Socal lstm: Human trajectory prediction in crowded spaces," In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVRP)*, pages 961-971, IEEE, 2016.

[11] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: distant future prediction in dynamic scenes with interacting agents," In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVRP)*, pages 2165-2174, IEEE, 2017.

[12] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A Hierarchical LSTM Model for Pedestrian Trajectory Prediction," In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1186-1194, IEEE, 2018.

[13] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," MIT press Cambridge, 1998.

[14] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 663-670, 2000.

[15] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, pages 1-8, 2004.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 9(8):1735-1780, 1997.

[17] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," In *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML)*, pages 729-736, 2006.

[18] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1433-1438, 2008.

[19] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML)*, pages 2829-2838, 2012.

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

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

[22] D. Kingma, J. Ba, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] N. J. Goodall, "Vehicle automation and the duty to act," In *Proceedings of the Twenty-first World Congress on Intelligent Transport Systems*, 2014.