Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories

Anonymous Author(s)
Affiliation
Address
email

Abstract:
Predicting the motion of a mobile agent from a third-person perspective is an important component for many robotics applications, such as autonomous navigation and tracking. With accurate motion prediction of other agents, robots can plan for more intelligent behaviors to achieve specified objectives, instead of acting in a purely reactive way. Previous work addresses motion prediction by either only filtering kinematics, or using hand-designed and learned representations of the environment. Instead of separating kinematic and environmental context, we propose a novel approach to integrate both into an inverse reinforcement learning (IRL) framework for trajectory prediction. Instead of exponentially increasing the state-space complexity with kinematics, we propose a two-stage neural network architecture that considers motion and environment together to recover the reward function. The first-stage network learns feature representations of the environment using low-level LiDAR statistics and the second-stage network combines those learned features with kinematics data. We collected over 30 km of off-road driving data and validated experimentally that our method can effectively extract useful environmental and kinematic features. We generate accurate predictions of the distribution of future trajectories of the vehicle, encoding complex behaviors such as multi-modal distributions at road intersections, and even show different predictions at the same intersection depending on the vehicle's speed.

Keywords: inverse reinforcement learning, trajectory prediction, neural networks

1 Introduction

Most autonomous navigation and tracking applications include interactions with other agents in the world. If a robot can accurately forecast the motion of external agents, it can plan much more intelligent behaviors, instead of having purely reactive interactions. For example, to avoid collisions with other drivers on a road, an autonomous car should be able to model future trajectories of other vehicles, and a robot manipulator that interacts with humans should anticipate a person's behavior to avoid dangerous situations.

Typically, the motion prediction problem is addressed using filtering-based methods [1], which use specific kinematic and observation models to estimate the states of the subject, such as position, orientation, and velocity. However, predicting the motion of an agent considering only kinematics is a simplistic model of the true behavior, which also depends on the surrounding environment. Although a human expert can manually design functions that model the agent's interactions with objects, this process is usually time-consuming and offers weak generalization to new environments.

Recently, imitation learning based approaches showed the possibility of learning the complex agent-environment interaction [2, 3]. In particular, inverse reinforcement learning (IRL) techniques can recover a complex reward structure using expert demonstrations, and offer higher robustness to generalization than manually-designing functions or supervised learning [4]. Although the first IRL reward structures were linear [5, 6, 7], recent work [3, 8] improved the reward model complexity.

Submitted to the 2nd Conference on Robot Learning (CoRL 2018). Do not distribute.

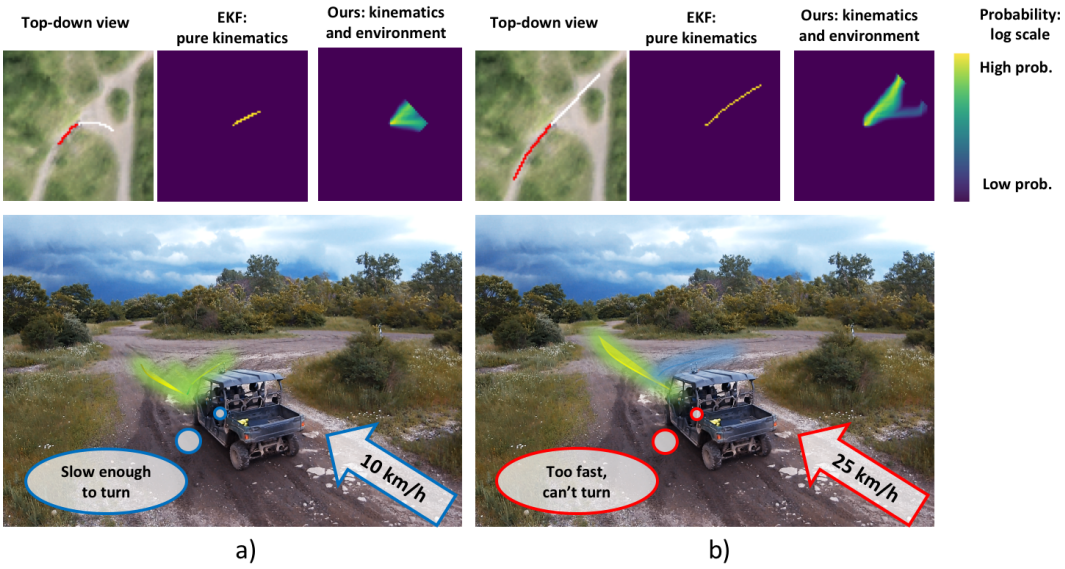

Figure 1: Trajectory prediction results on the same environment comparing different speeds. Slow speeds show stronger multi-modal prediction (a), while a fast vehicle is more likely to just continue going straight (b). The top-down view shows the trajectory of the past 5 s (red) and ground truth (white). We compare our method with a Kalman filter prediction.

A challenge in IRL comes from the choice of representation of the agent's state. Adding more dimensions to the state such as velocities and higher derivatives generally improves the fidelity of the model, but extra dimensions come with an exponential increase in computation complexity [9]. Finn [8] used importance sampling to reduce computation time when considering positions and velocities as the robot state. But in previous IRL work for vehicle prediction [3] does not consider kinematics when computing reward maps, using only position as the state.

Our key insight is that one can lower the exponential complexity increase of incorporating kinematics into the state-space if kinematic data is used instead during feature extraction. We offer three major contributions: 1) We propose an improvement on existing deep IRL frameworks, incorporating both kinematic and environmental context to predict trajectories from raw sensory input; 2) We train and verify the proposed method on a custom off-road driving dataset; 3) We qualitatively and quantitatively compare our prediction results with baseline methods such as extended Kalman filters and direct behavior cloning.

2 Related work

The problem of predicting the trajectory of an agent in the environment can be approached using different frameworks depending on the system's objective. A Kalman filter estimates linear and angular velocities, then forward-propagates the motion [1]. The filter ignores environmental context, and provides a uni-modal distribution over the space of possible trajectories. To model the agent's interactions with objects in the world, different methods manually design cost functions [10, 11, 12]. However, to overcome the issue of human fine-tuning and low generalization, one can explore the idea of learning behaviors from demonstrations.

Behavior cloning (BC) methods to model driving date as far back as 1989, when Pomerleau [13] used supervised learning to map the current first-person image from a car to the desired steering angle, and methods improved significantly over time [14, 15]. A drawback of BC is that in practice it is based on supervised learning with *non-i.i.d* samples due to the sequential aspect of data, therefore matching the training and test set distributions can be a challenge, hindering generalizability [4].

Inverse reinforcement learning (IRL) can be used to improve generalization of a policy. IRL's objective is to recover the reward function that the agent optimizes [4]. While early work on IRL

modeled rewards as linearly dependent on features [5, 6, 7, 2], more complex models followed [8, 16, 17, 18, 3].

Closest to our work, Wulfmeier et al. [3] use a neural network to model the mapping from sensor measurements coming from expert driving demonstrations to a reward map. A motion planner can then use the reward map to generate a path between desired start and goal locations. However, this approach ignores the role of dynamics in the forecast, considering only positions in the state-space. Adding extra dimensions to the state exponentially increases computational complexity for the task [9]. Thus, the training process slows down significantly and requires larger amounts of demonstrations for converge of a policy. Finn [8] attempts to overcome complexity by using importance sampling to compute state visitation frequencies from the learned policy. In this work, however, we argue that kinematics can instead be successfully used in the feature extraction process. Therefore, we propose a non-linear reward model that depends on both kinematics and environmental features, while keeping a small state-space dimension.

3 Approach

We frame the problem of vehicle trajectory prediction with max-entropy IRL. In order to incorporate both environment and kinematic context, a two-stage convolutional neural network (CNN) architecture is utilized to approximate the underlying reward function.

3.1 Problem formulation

Our objective is to learn to predict the target's future path ζ or probability distribution of future path $p(\zeta_i)$. We use a set of demonstrated trajectories $D = \{\zeta_0, \zeta_1, ...\zeta_m\}$, which are collected in various environments. The trajectory ζ is defined as a sequence of states $(s_0, s_1, ..., s_n)$, where $s = (x, y)$ represents the target's 2D location in world coordinates. Each state s is represented with features $\phi(s)$, obtained from sensory inputs. To predict the trajectory, one can directly estimate the state sequence, or predict the action sequence (a_0, a_1, a_2, a_3), where action a is the velocity of the agent in a 4-connected grid.

In behavior cloning, we can find a mapping from state and features to action: $\pi : s, \phi(s) \mapsto a$ [4]. In IRL, however, the goal is to recover the agent's hidden reward function $R(\phi(s); \theta)$, so as to maximize the probability of the demonstrated trajectories. Once the hidden reward is inferred, the probabilities of future trajectories can be computed.

3.2 Maximum entropy deep IRL

Here we follow the maximum entropy IRL formulation [7], treating trajectories with higher rewards as exponentially more likely. As shown by [3, 8], we approximate the reward function using a deep neural network $R(s; \theta) = f(\phi(s); \theta)$. Each trajectory ζ then has cumulative reward $R(\zeta) = \sum_{s_i \in \zeta} f(\phi(s_i); \theta)$. Under the maximum entropy assumption we have $P(\zeta_i|\theta) = \frac{1}{Z(\theta)} \exp \sum_{s \in \zeta_i} R(\phi(s); \theta)$, where $Z(\theta) = \sum_{\zeta_j} \exp(R(\zeta_j; \theta))$ is the partition function over all possible trajectories. Then we try to maximize the log likelihood of the demonstrated trajectories using its gradient [3]:

$$\mathcal{L}(\theta) = \log \prod_{\zeta_i \in D} P(\zeta_i; \theta) = \sum_{\zeta_i \in D} R(\zeta_i) - \log M \sum_{\zeta_j} \exp(R(\zeta_j; \theta)) \Rightarrow \frac{\partial \mathcal{L}(\theta)}{\partial \theta} = (\mu_D - \mathbb{E}[\mu]) \frac{\partial f}{\partial \theta}$$

Where μ_D and $\mathbb{E}[\mu]$ are the state visitation frequencies (SVF) from the demonstrated trajectories and from the current reward function respectively. After the reward network update, we solve the forward RL problem in the loop, as shown in Alg. 1. During value iteration, to speed up convergence we artificially increase the probability of the most likely action being chosen, in what we call *annealed softmax* in Alg. 2.

Algorithm 1 Deep maximum entropy IRL with two-stage network architecture

Input: D, S, A, T, γ, α
Output: network parameters θ

Initialize network parameters θ randomly
for iteration $i = 1$ to N **do**
  sample demonstration batch $D_{batch} \subset D$
  $R_i \leftarrow f(\phi(S); \theta_i)$ ▷ Forward reward network
  $\pi_i \leftarrow value\_iteration(R_i, S, A, T, \gamma)$ ▷ Planning step
  $\mathbb{E}[\mu_i] \leftarrow compute\_svf(\pi_i, S, A, T)$
  $\frac{\partial \mathcal{L}(\theta)}{\partial R} \leftarrow \mu_D - \mathbb{E}[\mu_i]$ ▷ Gradient calculation
  $\theta_{i+1} \leftarrow back\_propagate(f, \theta_i, \frac{\partial \mathcal{L}(\theta)}{\partial R}, \alpha)$ ▷ Parameter update
return θ

Algorithm 2 Value iteration

Input: R, S, A, T, γ
Output: π

Initialize values $V(s) = -\inf$
repeat
  $V_t(s) = V(s)$ ▷ No hard reset for $V(s_{goal})$
  $Q(s, a) = r(s, a) + E_{T(s,a,s')}[V(s')]$
  $V(s) = max(Q_i(s, a))$
until $max_s(V(s) - V_t(s)) < \epsilon$
return $\pi(a|s) = annealed\_softmax(Q(s, a))$

3.3 Incorporating kinematics into the reward: two-stage network architecture

Intuitively, when humans predict vehicle motion from a third-person perspective we primarily reason about two aspects: the surrounding environment and the vehicle's motion so far. We implicitly evaluate traversability of the terrain, the location of obstacles, and extrapolate the past vehicle motion to infer where turning will be necessary and/or possible. If given enough observations, we can even infer more subtle cues such as driver preferences and a specific motion model for the vehicle.

Inspired by this intuition, we designed a two-stage network architecture to reason about environmental and kinematic context when computing rewards (Figure 2). In the first stage, we adopt a four-layer fully convolutional network (FCN) [19] structure which takes LiDAR statistics and a top-down RGB image as inputs, and outputs features extracted purely from the environmental context. Instead of using encoder-decoder approaches [19, 20, 21], which inevitably lose spatial information

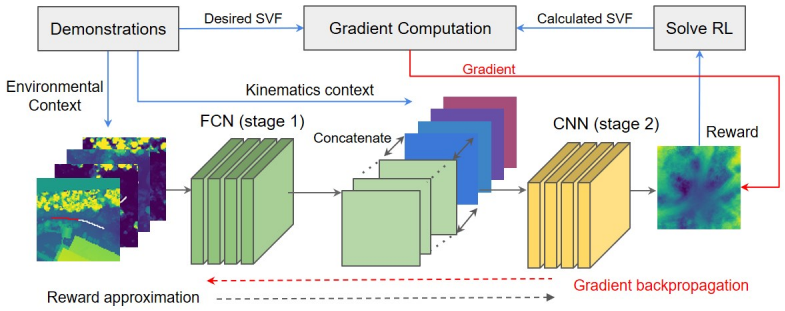

Figure 2: Proposed two-stage network architecture and training procedure. Environmental context is the input to the first-stage network, and the resulting feature maps are concatenated with the kinematics context. The reward approximation is the output of the second-stage network, and the state visitation frequency (SVF) from the learned policy is used to compute gradients for backpropagation.

due to the down-sampling stages, we adopt dilated convolutional layers, which systematically aggregate multi-scale context without losing spatial information [22]. The approximate receptive field of our dilated network is 20×20 pixels, which we estimated to be sufficient to extract the relevant environmental context. The final output of the dilated network is 25 feature maps, a number experimentally observed to contain sufficient information representing spatial context without excessive redundancy and sparsity.

As inputs to the second stage, we concatenate the output from the first stage with two feature maps encoding positional information and three feature maps representing kinematic information from the past trajectory, as follows. The first two feature maps encode, for each grid cell, the x and y position of the grid cell in a vehicle-centered, world-aligned frame. These feature maps are independent of the trajectories, but convey absolute position information to fully convolutional network, which is translation-invariant[1]. Then, for each training sample, past trajectory information is encoded with three feature maps $\phi_e(\zeta) = [\Delta x, \Delta y, \kappa]$. Δx and Δy represent the vehicle's past velocity, discretized to the four cardinal directions, and with a normalized magnitude proportional to its absolute speed. κ encodes the trajectory curvature, which is related to angular velocity. $[\Delta x, \Delta y, \kappa]$ are estimated from the past five seconds of vehicle motion and respectively constant across the whole feature map. Finally, they are empirically normalized to a small finite range to aid training stability.

A complete documentation about implementation details and network parameters is available in the supplementary material.

4 Experiments

In this section we present our approach for dataset collection, baseline design, definition of metrics, prediction results, and discussion on learned results.

4.1 Off-road driving dataset collection

The off-road driving dataset was collected in a test site with roughly 400 acres involving more than 5 different drivers. The vehicle platform, a modified All Terrain Vehicle (ATV) (Figure 3), is equipped with high precision RTK-GNSS/INS which can provide position data with centimeter level accuracy. A total of over 1000 trajectories are included for the dataset, with average length about 20-40 m long, in about 30 km of driving demonstrations.

Figure 3: Vehicle platform equipped with RTK GNSS, along with satellite view of the off-road course used for about 30 km of data collection.

For each demonstration we create a high-precision, colored local point cloud map. We then convert the point cloud into a 2D grid with statistics for each cell: maximum height, height variance, and mean RGB values, and extract the vehicle past trajectory and ground truth prediction using RTK GNSS.

The reason behind our choice of features is that in off-road environments, a combination of geometry and color statistics can provide useful information regarding terrain traversability. For example, one can intuitively categorize regions with high variance in height to be rough and non-traversable. However, combining geometry with color can be even more informative: an area that is simultaneously rough but light green is probably just a region of tall grasses, over which the vehicle can easily drive. An example of data from our dataset of demonstrations is shown in Figure 4.

[1] An alternative way of encoding absolute spatial position would be using a fully-connected network, but this will result in a substantial increase in the number of parameters.

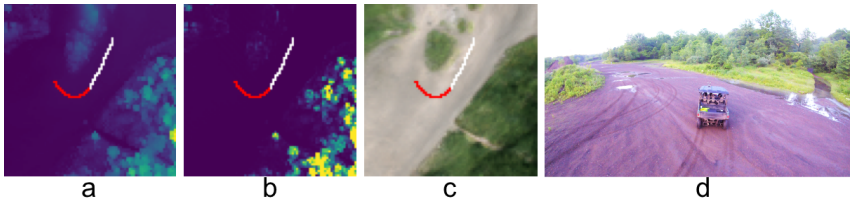

Figure 4: Visualization of all channels of the input to the reward network. a) Maximum height, b) Height variance, c) RGB, d) Third-person view of the environment (not an input). Red indicates the trajectory history, which is used to compute kinematic features, and ground truth future path is shown in white.

4.2 Metrics and baselines

We employ two metrics for quantitative evaluation: the Negative Log-Likelihood (NLL) of the demonstration data and the Hausdorff Distance (HD) [2]. The NLL metric computes the log-likelihood of the demonstration trajectory ζ under the learned policy $\pi(a|s)$. We normalize NLL by the demonstration trajectory length. The HD metric represents a spatial similarity between expert demonstrations and trajectories sampled with the learned policy. To compute it, we use the average HD between the demonstration trajectory and 1000 trajectories randomly sampled from the learned policy.

We selected three baseline methods for comparison: 1) Extended Kalman filter (EKF) with a kinematic bicycle model; 2) behavior cloning technique that uses supervised learning to learn a policy mapping the state and features to an action; and 3) deep IRL method considering only environment input to compute features, with no kinematics.

4.3 Trajectory prediction results

Qualitative evaluation

We show key experimental results in Figure 5, and a graphical comparison with baselines in Figure 6. More visualizations of the experiments can be found in the supplementary video. We verify that the forecasts calculated based on the inferred reward map, match the expected behaviors of a human driver, with trajectory probabilities concentrated on trails or traversable paths. In addition, multi-modal behaviors are present at intersections and when approaching open areas.

In Fig. 1 we see how the trajectory prediction behaves on the same environments, for different agent kinematics. Our predictions are surprisingly intuitive: at lower speeds the forecast shows higher

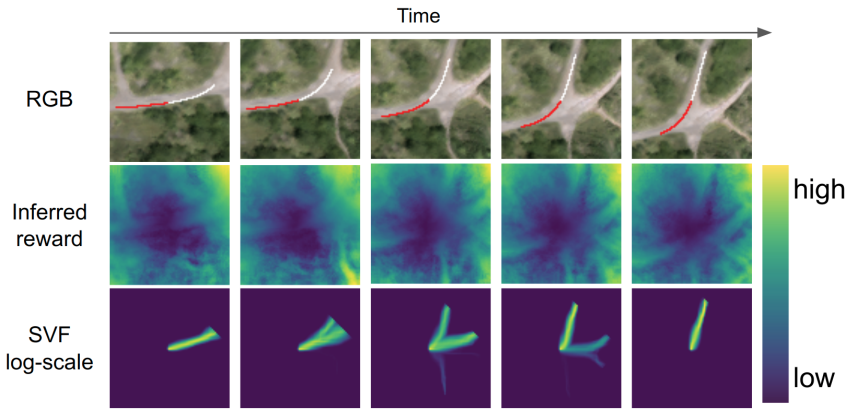

Figure 5: Time-lapse of inferred reward map and trajectory forecast distributions in an off-road trail test set. Forecasts remain on trails, and show multi-modal distributions at intersections.

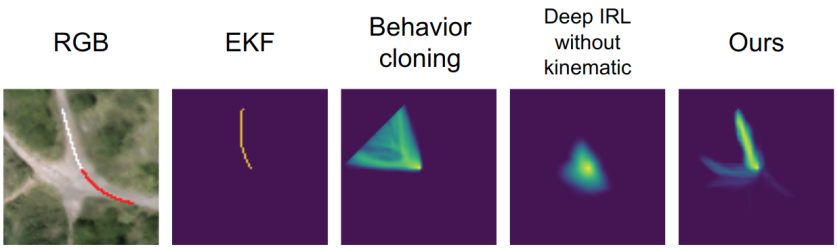

Figure 6: Comparison of trajectory predictions for different baselines in test set. The kalman filter ignores any environmental context, and behavior cloning fails to learn a policy that avoids obstacles. Without kinematic features, deep IRL has no notion of where the vehicle is going, and just forecasts a diffuse motion on the large open section of trail around the actor's position. Our method is able to capture both kinematics and environmental cues, inferring multiple plausible paths.

likelihood of taking turns at an intersection, while at high speeds it predicts that the vehicle will probably keep driving straight along the trail.

Quantitative evaluation

We compare our method with four baselines in the off-road driving dataset, shown in Table 1. Our method has the best prediction results on the test set using both metrics. We observe that the EKF obtains surprisingly good results for HD; we believe that this is due to a relatively large proportion of straight or near-straight trajectories in the dataset, which EKF can predict well. We also observe a comparatively poor performance of Deep IRL without kinematic features. Examination of the predictions suggests that without past kinematic context, in many open scenarios this method can only predict diffuse, highly uncertain future motions, as shown in Fig. 6. On the other hand, our method, which can exploit both environmental and kinematic context, obtains the best results.

Method	EKF	Behavior cloning	Random	Deep IRL without kinematics	Ours
NLL	N.A.	0.87	1.35	1.33	**0.69**
HD	9.12	10.54	25.62	25.46	**6.71**

Table 1: Prediction performance comparison on the test set using NLL and HD. For both NLL and HD, lower numbers represent better predictions. Our method obtains the best results in both evaluation metrics.

4.4 What is learned?

We try to build an understanding of what is being learned by plotting representative outputs from different stages of the network in Figs. 7-8. The first-stage is network apparently encoding the traversable trail based on LiDAR input. As seen in Fig. 8, the second stage of the network learns an effective forward motion model of the car with an uncertainty cone roughly oriented towards the direction of motion.

5 Discussion and future work

In this work we proposed and validated a deep IRL approach that integrates kinematic and environmental context to learn a reward structure for driving predictions in off-road environments. We show that our approach out-performs baselines such as Kalman filtering, behavior cloning, and a deep IRL approach without considering kinematics.

Our two-stage network architecture used for the reward approximation can mitigate the computational complexity of adding kinematics as extra dimensions to the state-space; we instead add kinematics in the feature extraction process. Based on our findings, we see future work in a few directions:

Improve kinematics model: Our kinematic features come from a constant 5 s window from the past trajectory. When the driver does a sudden motion, using a fixed time window is not ideal, because

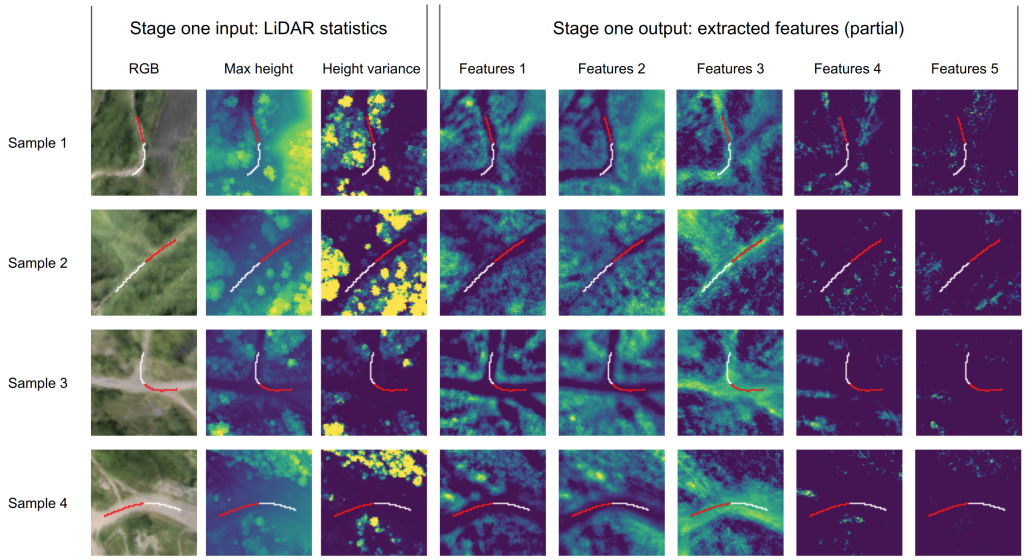

Figure 7: Visualization of select feature maps from the first-stage network output. Features 1-3 have dense output patterns that visibly correlate to the trail region. Features 4-5 show a relatively sparse output, possibly encoding specifiic LiDAR statistics.

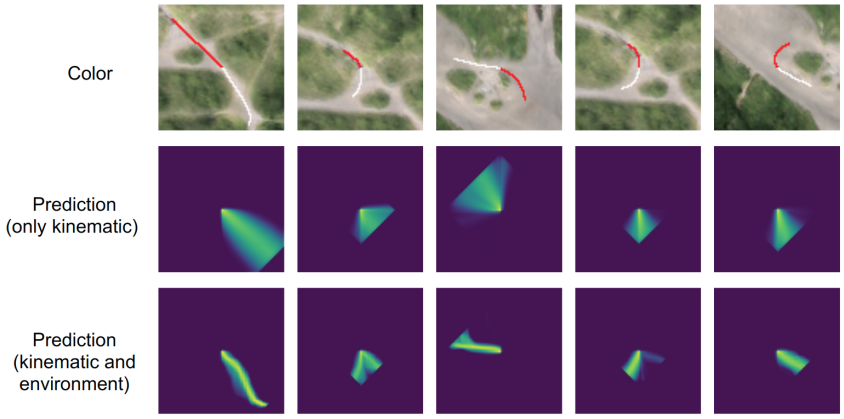

Figure 8: Visualization of trajectory probabilities with and without environment context. The first row shows past trajectories (red) and ground-truth (white). The second row shows predictions using our method where LiDAR features were replaced by a constant value in all cells, and we can compare with predictions that incorporate both kinematics and environment in the third row. The network learns a forward motion model of the car with an uncertainty cone roughly oriented towards the direction of motion, and that varies with respect to vehicle speed.

under sudden movements only a smaller time segment should be considered to forecast where the vehicle will go. We envision improvements in the second-stage network, where the network can learn how to select adaptive time windows for prediction, probable using recurrent-neural-networks.

Continuous states and actions: Although the grid-based model provided sufficient motion resolution for our particular application, in many robotics problems it is necessary to define continuous states and actions.

Cameras as sensor input: Cameras are the most common type of sensor in robots, and image data contains rich information about the environment. We envision using images, together with kinematics, for the CNN-based feature extraction.

References

[1] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[2] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.

[3] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[4] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[5] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[6] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

[7] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[8] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[9] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[10] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[11] 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.

[12] C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI magazine*, 30 (2):17, 2009.

[13] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[14] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2722–2730. IEEE, 2015.

[15] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[16] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[17] A. Grubb and J. A. Bagnell. Boosted backpropagation learning for training deep modular networks. In *ICML*, pages 407–414, 2010.

[18] S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

[19] 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.

[20] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[21] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[22] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.