Improved Activity Forecasting for Generating Trajectories

Daisuke Ogawa[1], Toru Tamaki[1], Tsubasa Hirakawa[2], Bisser Raytchev[1], Kazufumi Kaneda[1], Ken Yoda[3]
[1]Hirohsima University, Japan, [2]Chubu University, Japan, [3]Nagoya Univerisity, Japan

***Abstract*—An efficient inverse reinforcement learning for generating trajectories is proposed based of 2D and 3D activity forecasting. We modify reward function with L_p norm and propose convolution into value iteration steps, which is called convolutional value iteration. Experimental results with seabird trajectories (43 for training and 10 for test), our method is best in terms of MHD error and performs fastest. Generated trajectories for interpolating missing parts of trajectories look much similar to real seabird trajectories than those by the previous works.**

***Index Terms*—inverse reinforcement learning, trajectory interpolation, trajectory generation**

I. Introduction

Analyzing, understanding and predicting human and animal movement in forms of trajectory is an important task [1]–[3] and has been studied in computer vision [4], [5] as well as ecology [6], [7]. Despite of recent recent progress in deep learning [8]–[11], those methods are hard to apply to ecological data because of small size of datasets; at most few dozens of trajectories are available because it is not an easy task to obtain trajectory data of animals.

In this paper we propose an improvement of activity forecasting [12] that predicts distributions of trajectories over a 2D plane by using maximum entropy (MaxEnt) based [13] inverse reinforcement learning (IRL) [14]. Activity forecasting has been applied to trajectories of birds by extending the state space from 2D to 3D, in order to simulate bird behavior by generating trajectories [6] and interpolate missing parts in trajectories [7]. However the computation cost in both time and space is very demanding because of the 3D extension. We investigate formulations and implementations of previous works, derive an improvement and efficient formulation, and present a stochastic method for generating trajectories that are more similar to real trajectories than those by the previous deterministic approaches.

II. Methods

A. Reinforcement and inverse reinforcement learning

Reinforcement learning (RL) [15], [16] formulates a problem in which agents in an environment take actions for maximizing rewards, and modeled by Markov Decision Process (MDP). The agent gains immediate reward r_t at state s_t by choosing action a_t at time step t, then move to the next state according to state transition probability p. Through a learning procedure, we obtain a policy π that maximizes the accumulated reward R through a series of actions. For a small discrete problem in 2D, the state space is usually discretized as a grid, and a trajectory is represented by a series (or trajectory) $\xi = \{s_1, \ldots, s_t\}$ of 2D grid coordinates $s_t = (x_t, y_t)$, where x_t and y_t are two-dimensional grid coordinates. Actions a_t can be defined as moves to neighboring eight grid states.

A common approach to RL is Q-leaning [16] that iteratively update action value function $Q(s_t, a_t)$ with weight α and discount factor γ by

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')), \tag{1}$$

that approximates the Bellman equation after convergence. Then the greedy policy taking the action that maximizes Q at s_t, is optimal.

Reward values are however not always given or difficult to define in practice, therefore inverse Reinforcement Learning (IRL) [14] is used to estimate the reward values, and the maximum entropy (MaxEnt) approach [13] formulates the IRL problem as follows.

B. MaxEnt IRL

The reward value of trajectory ξ is expressed by linear combination of feature vector $\boldsymbol{f}$,

$$R(\xi; \boldsymbol{\theta}) = \sum_t r(s_t; \boldsymbol{\theta}) = \sum_t \boldsymbol{\theta}^T \boldsymbol{f}(s_t), \tag{2}$$

where $\boldsymbol{\theta}$ is the parameter to be estimated, and $\boldsymbol{f}(s_t)$ is a pre-defined feature vector value at s_t. This approach defines probability of trajectory ξ by[1]

$$p_\pi(\xi|\boldsymbol{\theta}) \propto \exp(R(\xi; \boldsymbol{\theta})) = \exp\left(\sum_t \boldsymbol{\theta}^T \boldsymbol{f}(s_t)\right) \tag{3}$$

and solve the following log-likelihood maximization problem;

$$\operatorname*{argmax}_{\boldsymbol{\theta}} L(\theta) = \operatorname*{argmax}_{\boldsymbol{\theta}} \frac{1}{|Z|} \sum_{\xi \in Z} \log p_\pi(\xi|\boldsymbol{\theta}), \tag{4}$$

where Z is a given set of trajectories. The weight vector $\boldsymbol{\theta}$ is updated by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \mathrm{e}^{\lambda \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})}, \tag{5}$$

until convergence with leaning late λ. The gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \overline{f} - E_{p_\pi(\xi'|\boldsymbol{\theta})}\left[\sum_t \boldsymbol{f}(s'_t)\right], \tag{6}$$

[1]The denominator (partition function) has θ and hence should be considered (but omitted here for simplicity).

Algorithm 1 Backward-forward algorithm [12], [13]

(Backward pass)
$V(s) \leftarrow -\infty$
for $n = N$ to 1 **do**
  $V^{(n)}(s_{goal}) \leftarrow 0$
  $Q^{(n)}(s,a) = r(s;\boldsymbol{\theta}) + E_{p(s'|s,a)}[V(s')]$
  $V^{(n-1)}(s) = \mathrm{softmax}_a\, Q^{(n)}(s,a)$
end for
$\pi_{\boldsymbol{\theta}}(a|s) \propto \exp(Q(s,a) - V(s))$

(Forward pass)
$D(s_{initial}) \leftarrow 1$
for $n = 1$ to N **do**
  $D^{(n)}(s_{goal}) \leftarrow 0$
  $D^{(n+1)}(s) = \sum_{s',a} p(s'|s,a)\pi_{\boldsymbol{\theta}}(a|s')D^{(n)}(s')$
end for
$D(s) = \sum_n D^{(n)}(s)$

where $\overline{f} = \frac{1}{|Z|}\sum_{\xi\in Z}\sum_t \boldsymbol{f}(s_t)$. Note that the expectation in the second term is taken for all possible path ξ' according to policy π, which is intractable, and is approximated by the following weighted sum for given trajectories Z,

$$E_{p_\pi(\xi'|\boldsymbol{\theta})}\left[\sum_t \boldsymbol{f}(s_t)\right] \approx \frac{1}{|Z|}\sum_{\xi\in Z}\sum_t \boldsymbol{f}(s_t)D_\xi(s_t), \quad (7)$$

where $D_\xi(s_t)$ is expected state frequency.

C. Activity forecasting in 2D

Activity forecasting [12] utilized the MaxEnt IRL approach for dealing with noisy pedestrian trajectories by implicitly introducing hidden states. Action-value function Q and state-value function V are defined as follows;

$$Q(s,a) = r(s;\boldsymbol{\theta}) + E_{p(s'|s,a)}[V(s')] \quad (8)$$
$$V(s) = \operatorname*{softmax}_a Q(s,a), \quad (9)$$

where $p(s'|s,a)$ is state transition probability, and softmax is a soft version of the maximum that is defined as $\mathrm{softmax}_a\, Q(s,a) = \max_a Q(s,a) + \log[1 + \exp\{\min_a Q(s,a) - \max_a Q(s,a)\}]$ in their implementation[2]. The policy π is then defined by

$$\pi(a|s,\boldsymbol{\theta}) \propto \exp(Q(s,a) - V(s)). \quad (10)$$

The backward-forward algorithm is shown in Algorithm 1 that compute the policy and $D(s)$.

D. Activity forecasting in 3D

Hirakawa et al. [7] extended the 2D activity forecasting to 3D for dealing with time explicitly to achieve accurate interpolation of missing parts in GPS trajectories of seabirds. They extend two-dimensional states to three-dimensional states by augmenting time step; a state is defined as $s_t = (x_t, y_t, z_t)$, where x_t and y_t are 2D grid coordinates and z_t is a discrete time step. This is because seabirds may take an indirect route between two points, while IRL methods including 2D activity forecasting tend to produce direct routes. In this work, an action is defined as $a = (a_{xy}, 1)$, where a_{xy} is moves to neighboring eight grid states in 2D, and the last augmented value of one enforces the increment in time step when taking one action. Now a trajectory is denoted by $\xi = \{(s_0,a_0),(s_1,a_1),\dots\}$. Because of this 3D extension, their 3D method performs better than the original 2D forecasting, at the cost of increased computation time and required memory. This cost is very demanding and therefore efficient improvements would be necessary.

E. Proposed method

To achieve a better performance with a smaller computation cost, we propose the following improvements.

First, we take 2D approach like as the original activity forecasting [12]. A 3D extension approach [7] may work better, but inherently increases computation cost because of three-dimensional state space.

Second, we "exactly" follow the definitions of Q and V as shown in Eqs. (8) and (9). Implementations of previous works[3] are different from the equations, and evaluate the softmax function for eight actions. This is one of the most computationally intensive part in the implementations. The use of softmax is introduced in [13], but in preliminary experiments we found that softmax doesn't affect results so much and can be replaced with max as in common value iteration.

Third, we make reward values dependent to next state s' as well as current state s to imitate the effect of the original implementation of 2D activity forecasting. It is natural to move every direction in 2D with the same cost under the same condition, however the eight moves in neighboring 2D grid state result in the preference to diagonal moves. For example, one step to north-west from the current state is longer than a step to the west, but the reward is a function of current state, $r(s)$. Therefore we propose to define the reward as $r(s,s',a)$ in order to take the distance between adjacent states. More speficically,

$$r(s,s',a) = r(s)/\mathrm{dist}_p(s,s'), \quad (11)$$

where $\mathrm{dist}_p(s,s')$ is L_p distance between states s and s'. In experiments, we compare results with $p = 2$ and $p = 3$ because $p = 2$ is the Euclidean distance, which seems to be natural in this task, and $p = 3$ produces results similar to the effect by the original 2D activity forecasting implementation.

Fourth, we propose an effective value iteration called *convolutional value iteration*. The core of value iteration is to iterate Eq.(8) and Eq.(9) to propagate values. Here, we insert a convolution step with the Gaussian kernel G to make V blur, which effectively accelerate the propagation and convergence

[2] http://www.cs.cmu.edu/~kkitani/datasets/

[3] http://www.cs.cmu.edu/~kkitani/forecasting/code/oc.cpp for [12] and https://github.com/thirakawa/MaxEnt_IRL_trajectory_interpolation for [7].

TABLE I

MHD OF INTERPOLATION RESULTS AND COMPUTATION TIME. AVERAGE VALUES WITH STD ARE REPORTED. $p = 2$ OR $p = 3$ INDICATES L_p NORM USED IN THE MODIFIED REWARD FUNCTION. W/ OR W/O INDICATES CONVOLUTIONAL OR ORDINAL VALUE ITERATION. NOTE THAT COMPUTATION TIME OF VALUE ITERATION IS FOR A SINGLE ITERATION OF EQS.(8) AND (9), AND UPDATE OF θ IS FOR COMPUTATION OF $\nabla_\theta L(\theta)$ FOR A SINGLE UPDATE. COLUMNS OF RATIO IS BASED ON ROW "W/O CONV".

	MHD (deterministic)	MHD (stochastic)	value iteration [s]	ratio	update of θ [s]	ratio
Linear	12.20 ± 4.48					
3D [7]	5.33 ± 2.61		4.0 ± 0.19	1191	22354 ± 449	2064
2D [12]	5.80 ± 2.96	6.02 ± 2.97	0.008667 ± 0.003229	2.57	15.05 ± 0.78	1.39
$p = 2$ w/o conv	6.37 ± 3.19	$\mathbf{5.13 \pm 2.77}$	$\mathbf{0.003367 \pm 0.002389}$	1	$\mathbf{10.83 \pm 0.85}$	1
$p = 3$ w/o conv	6.14 ± 3.35	5.20 ± 3.09				
$p = 2$ w/ conv	6.45 ± 3.36	5.22 ± 2.82	0.003542 ± 0.002242	1.05	11.5608 ± 0.8293	1.07
$p = 3$ w/ conv	5.63 ± 3.32	5.20 ± 3.06				

of iteration. The proposed method has the following form of iteration;

$$Q(s,a) = E_{p(s'|s,a)}[r(s,s',a;\boldsymbol{\theta}) + V(s')] \tag{12}$$

$$V(s) = (\max_a Q(s,a)) \otimes G \tag{13}$$

Fifth, we define policy π by

$$\pi(a|s,\boldsymbol{\theta}) \propto \exp(Q(s,a)), \tag{14}$$

which is more common in RL than Eq.(10).

F. Trajectory generation

Once a policy has been obtained, a trajectory can be generated either deterministic or stochastic way. A deterministic trajectory interpolation is to obtain next states by repeatedly selecting the action that maximize the policy at current states. A stochastic interpolation instead selects an action by sampling according to the policy. We compare two ways; [7] generated deterministic trajectories, however those look rather straight compared to real trajectories. Stochastic trajectories are expected to be more realistic.

III. EXPERIMENTAL RESULTS

The experimental setting is the same with [7]; we used 43 trajectories for training and 10 trajectories for test. and each test trajectory has missing parts to be interpolated. We compared our proposed method with linear interpolation, activity forecasting in 2D [12] and 3D [7].

The modified Hausdorff distance (MHD) [17] was used as a metric for quantitative evaluation. MHD is used to compute the similarity between object shapes. Given two trajectories, or two series of points $\boldsymbol{A} = \{a_1, \ldots, a_{N_a}\}$ and $\boldsymbol{B} = \{b_1, \ldots, b_{N_b}\}$, MHD between $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined by

$$\mathrm{MHD}(\boldsymbol{A},\boldsymbol{B}) = \max\left\{ \frac{1}{N_a}\sum_{a\in\boldsymbol{A}} d(a,\boldsymbol{B}), \frac{1}{N_b}\sum_{b\in\boldsymbol{B}} d(b,\boldsymbol{A}) \right\}, \tag{15}$$

where $d(a,\boldsymbol{B}) = \min_{b\in\boldsymbol{B}} ||a-b||$.

Table I shows experimental results. In terms of MHD, results of our proposed method with $p = 2$ or $p = 3$ with or without convolutional value iteration are not better than the previous works when deterministic trajectories are used. However with stochastic trajectory generation, our method works better than 2D activity forecasting. Convolutional value iteration was not observed to be effective for producing stochastic trajectory generation, while it may help for a faster convergence. Note that we exclude results of stochastic version of 3D approach because it doesn't guarantee to generate trajectories ending at the given time with stochastic policy sampling.

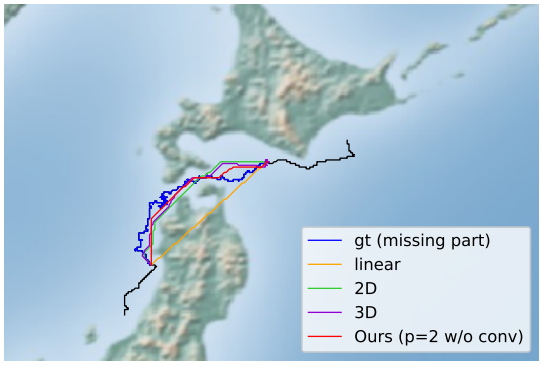

Fig. 1. Deterministic interpolation of the missing part in a trajectory by the proposed method and baseline methods. "gt" is ground-truth of the trajectory.

In terms of computation cost, our method performs much faster than 3D approach [7], which is more than factor of 1000, and even faster than the original 2D approach [12]. Note that all implementations were written in python and evaluated on the same computer. Required memory by our method is also much smaller than 3D approach, and almost similar to 2D approach (not reported here).

Figure 1 shows an example of deterministic trajectory interpolation with different methods. Due to the nature of deterministic trajectory generation, results by all methods look similar and straight vertically, horizontally, or diagonally. Figure 2 shows stochastic interpolation results of five different runs. Results of 2D and 3D approaches still looks straight, which may caused by estimated policies that assign larger probabilities to one single action at each state. In comparison, our method succeeded to generate more realistic trajectories. This might be attributed to the modified reward and policy.

Figures 3 and 4 show results of our method with different parameters. Interpolation results look similar to each other and therefore parameters doesn't affect results.

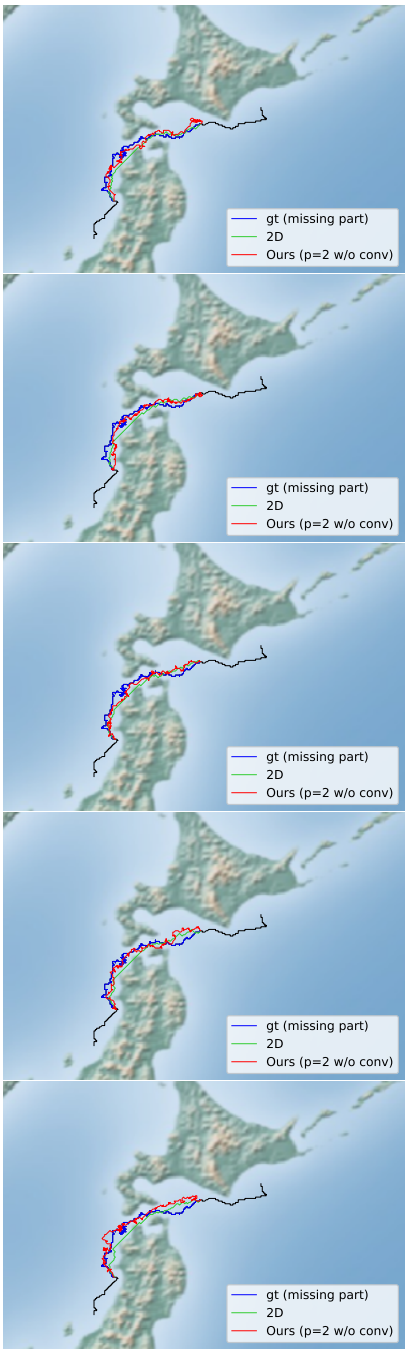

Fig. 2. Five different results of stochastic interpolation of the missing part in a trajectory by the proposed method and 2D approach. "gt" is ground-truth of the trajectory.

IV. Conclusion

We have proposed an efficient inverse reinforcement learning for generating trajectories based on 2D and 3D activity forecasting. Experimental results with a real dataset demonstrated that the proposed method works faster than 3D approach and effective for generating realistic trajectories. Future work includes handling temporal information. We have chosen a 2D approach, however time is still an important factor for interpolation and generation of trajectories. Keeping computation cost small and adding temporal factor is an challenging problem.

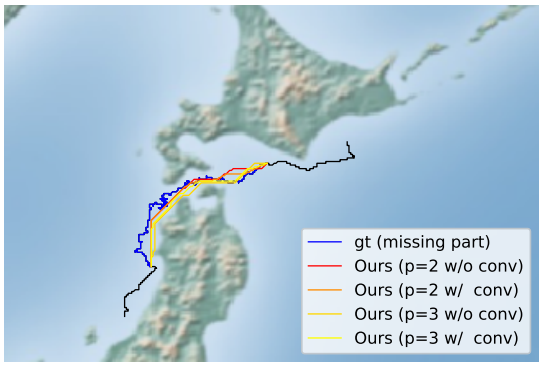

Fig. 3. Deterministic interpolation of the missing part in a trajectory by the proposed method with different parameters. "gt" is ground-truth of the trajectory.

Acknowledgments

This work was supported by JSPS KAKENHI grant numbers JP16H06540, JP16K21735, and JP16H06541.

References

[1] Brendan Tran Morris and Mohan Manubhai Trivedi. Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2287–2301, Nov 2011.

[2] Teng Li, Huan Chang, Meng Wang, Bingbing Ni, Richang Hong, and Shuicheng Yan. Crowded scene analysis: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(3):367–386, March 2015.

[3] Tsubasa Hirakawa, Takayoshi Yamashita, Toru Tamaki, and Hironobu Fujiyoshi. Survey on vision-based path prediction. In *Distributed, Ambient and Pervasive Interactions: Technologies and Contexts - 6th International Conference, DAPI 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part II*, pages 48–64, 2018.

[4] Bolei Zhou, Xiaoou Tang, and Xiaogang Wang. Learning collective crowd behaviors with dynamic pedestrian-agents. *International Journal of Computer Vision*, 111(1):50–68, 2015.

[5] Daisuke Ogawa, Toru Tamaki, Bisser Raytchev, and Kazufumi Kaneda. Semantic segmentation of trajectories with agent models. In *The International Workshop on Frontiers of Computer Vision (FCV2018)*, 2018.

[6] Tsubasa Hirakawa, Takayoshi Yamashita, Ken Yoda, Toru Tamaki, and Hironobu Fujiyoshi. Travel Time-dependent Maximum Entropy Inverse Reinforcement Learning for Seabird Trajectory Prediction. In *Asian Conference on Pattern Recognition*, 2017.

[7] Tsubasa Hirakawa, Takayoshi Yamashita, Toru Tamaki, Hironobu Fujiyoshi, Yuta Umezu, Ichiro Takeuchi, Sakiko Matsumoto, and Ken Yoda. Can ai predict animal movements? filling gaps in animal trajectories using inverse reinforcement learning. *Ecosphere*, 9(10), 2018.

[8] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Pedestrian behavior understanding and prediction with deep neural networks. In *European Conference on Computer Vision*, pages 263–279. Springer, 2016.

[9] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[10] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *Neural networks*, 108:466–478, 2018.

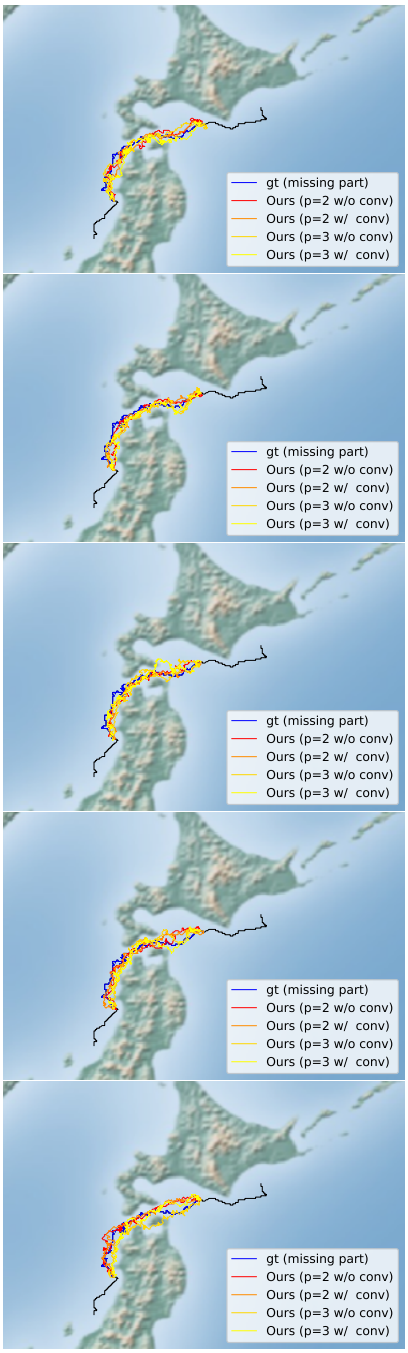

Fig. 4. Five different results of stochastic interpolation of the missing part in a trajectory by the proposed method with different parameters. "gt" is ground-truth of the trajectory.

[11] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher Bongsoo Choy, Philip H. S. Torr, and Manmohan Krishna Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. *CoRR*, abs/1704.04394, 2017.

[12] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.

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

[14] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[15] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[17] M-P Dubuisson and Anil K Jain. A modified hausdorff distance for object matching. In *Proceedings of 12th international conference on pattern recognition*, pages 566–568. IEEE, 1994.