Learning How Pedestrians Navigate: A Deep Inverse Reinforcement Learning Approach

Muhammad Fahad, Zhuo Chen, and Yi Guo

***Abstract*— Humans and mobile robots will be increasingly cohabiting in the same environments, which has lead to an increase in studies on human robot interaction (HRI). One important topic in these studies is the development of robot navigation algorithms that are socially compliant to humans navigating in the same space. In this paper, we present a method to learn human navigation behaviors using maximum entropy deep inverse reinforcement learning (MEDIRL). We use a large open dataset of pedestrian trajectories collected in an uncontrolled environment as the expert demonstrations. Human navigation behaviors are captured by a nonlinear reward function through deep neural network (DNN) approximation. The developed MEDIRL algorithm takes feature inputs including social affinity map (SAM) that are extracted from human motion trajectories. We perform simulation experiments using the learned reward function, and the performance is evaluated comparing it with the real measured pedestrian trajectories in the dataset. The evaluation results show that the proposed method has acceptable prediction accuracy compared to other state-of-the-art methods, and it can generate pedestrian trajectories similar to real human trajectories with natural social navigation behaviors such as collision avoidance, leader-follower, and split-and-rejoin.**

I. Introduction

Robots have been increasingly used in spaces where they cohabit with humans, such as museums [1], supermarkets [2], and offices [3]. An important task that the robot must perform while coexisting with humans in these environments is navigation, where the robot must navigate in close proximity to pedestrians. While navigating in such a pedestrian occupied environment, robots need to be able to comprehend and comply with typical social norms that pedestrians follow and maintain a comfortable distance from each other [4]. To achieve this, we need to model the behavior of pedestrians navigation in a crowded environment. This model can then be used by a robot to navigate while conforming to desired characteristics such as comfort, naturalness and sociability [5]. It is important to learn the cooperative pedestrian-pedestrian interaction and to incorporate this cooperative navigation for robot motion planning in human environments. In this paper, we focus on learning how humans navigate in crowds, and propose a deep inverse reinforcement learning (IRL) method in order to model pedestrian navigation behaviors.

Traditional socially compliant robot navigation can be broadly categorized into *model based* and *machine learning based* approaches. Gonzalo et. al. present a social force model (SFM) [6] based approach for human aware navigation [7]. An important aspect in the SFM based approaches is the appropriate selection of SFM parameters since these parameters need to be tuned for different environment conditions [8]. Hanheide et. al. adopt qualitative trajectory calculus (QTC) based approach for the analysis and representation of spatial behavior governing the interactions between a pedestrian and a robot [9]. A dynamic window approach for reactive collision avoidance for mobile robots is presented by Fox et. al. in [10]. The main focus of that approach is safety, since it is derived directly from the motion dynamics of the robots and is well suited for robots operating at higher speeds. A recent review of existing approaches can be found in [5]. The main shortcoming of the model based approaches is sensitivity to model parameter selection [8] and oscillatory, freezing and unsafe robot behavior when complexity exceeds certain degrees [4], [11].

Machine learning based approaches are receiving increased attention recently in robot social navigation research. A framework for socially adaptive path planing in a dynamic environment for an autonomous wheel chair was proposed in [12] using Bayesian IRL, where training data were obtained by using a person to manually drive the wheel chair, while moving in the desired navigation scenarios. Henrik et. al. present a maximum entropy IRL based approach in [4] for predicting pedestrian trajectories inside a controlled environment with navigation between fixed points. These works consider limited training data scenarios [4] and linear reward function structures [4], [12]. Chen et. al. propose a reinforcement learning based approach for indoctrinating certain behaviors using careful design of a handcrafted reward function [13]. Alahi et. al. formulate the problem of trajectory prediction as a sequence generation task, and adopt a long short term memory (LSTM) approach to predict the future trajectory of people based on their past positions [14]. A comparison of the performance of different features used for robot navigation in the IRL framework was provided by Vasquez et. al. [15] using pedestrian navigation data obtained in a simulated environment.

In this paper, we use maximum entropy deep inverse reinforcement learning (MEDIRL) to learn the reward function that captures the behaviors exhibited by a pedestrian while navigating between its initial and goal positions. We adopt this approach since the pedestrian navigation behavior draws from many complex experiences, thus it is challenging to manually craft a reward function that captures all the

The work was partially supported by the US National Science Foundation under Grant CMMI-1527016.

Muhammad Fahad, Zhuo Chen and Yi Guo are with Department of Electrical & Computer Engineering Stevens Institute of Technology, Hoboken, NJ 07030, USA. {mfahad, zchen39, yguo1}@stevens.edu

behaviors exhibited by a pedestrian as done in [13]. We use the concept of *social affinity* that formalizes the interaction of a pedestrian with its neighboring pedestrians, and propose a deep neural network structure to approximate a nonlinear function representation of input feature vectors. We use an open pedestrian dataset collected in an uncontrolled mall environment for training the reward function network [16]. This dataset offers diverse data with varying pedestrian density conditions and various interaction scenarios, which aids in generalizing the learned reward functions. We show in the performance evaluation section that the learned reward function captures complex pedestrian interaction behaviors and reliably recreates pedestrian trajectories. This learned reward function can be used to control a robot to navigate in a pedestrian occupied environment in a socially complaint manner, which is in the scope of our future research.

The rest of the paper is organized as follows. Section II provides the formal problem formulation. The proposed solution algorithm, i.e., the deep IRL approach, is presented in Section III. The experiments for training our reward network are detailed in Section IV. Section V provides performance evaluation capturing pedestrian behaviors and replicating measured trajectories, which is followed by conclusions and future work in Section VI.

II. Problem Formulation

In this section, we first review the Markov Decision Process (MDP) and IRL, and then, we define our pedestrian motion modeling problem in an IRL framework.

A. Markov Decision Process (MDP)

An MDP is a probabilistic sequential decision making process. Each decision, more formally action, in an MDP setting depends only on the current state and is thus memoryless. A finite state MDP can be defined as a tuple (S,A,T,γ,R), where $S = \{s_1,\ldots,s_O\}$ denotes a finite set of O states called the **state space**. $A = \{a_1,\ldots,a_L\}$ denotes a finite set of L possible actions called the **action space**. T denotes a set of **state transition probabilities**, describing the evolution of a previous state s_{i-1} to a new one s_i, when executing a given action a_{i-1}. $\gamma \in [0,1)$ denotes the **discount factor**. R denotes the **reward function** that depends on state and actions.

In the MDP framework, the objective of reinforcement learning is to find the policy π, defined as a mapping π: $S \rightarrow A$, which gives the optimal action for each state that should be executed to maximize the expected reward. While the reinforcement learning method uses a given reward function on an MDP model to generate an optimal policy mapping from system observations to actions, IRL deals with the inverse problem of finding the reward from either an existing policy, an action-state sequence, or demonstrated state sequences.

B. Pedestrian Motion Model

In this work, we model the pedestrian as an agent whose decision making process, follows an MDP while navigating in a crowded environment. The pedestrian is assumed to navigate in a discretized grid based workspace as illustrated in Fig. 1. At any time instant, from its initial state s_1, the pedestrian can choose to take an action from the set of possible actions $\{a_0,a_1,a_2,...,a_8\}$, and end up in any of the adjoining states $\{ns_1,ns_2,...,ns_8\}$ as shown in Fig. 1. Thus the complete trajectory of the pedestrian from its initial state s_1 to its goal state s_K is a sequence of state and action pairs represented by $\zeta = \{(s_1,a_1),(s_2,a_2),\ldots,(s_K,a_K)\}$. We assume the pedestrian to be a deterministic MDP agent, necessitating any action a from state s will result in the agent reaching the next state s' with probability one. The selected action depends on the reward function R.

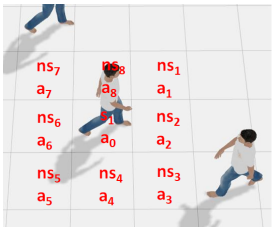


Fig. 1: Pedestrian motion model in a grid based workspace with adjoining states $ns_1, ns_2, ..., ns_8$ and corresponding actions $a_1, a_2, ..., a_8$ to reach those states.

In typical MDP problems, reward function R is hand crafted to indoctrinate the desired behaviors in an agent. In the pedestrian modeling problem studied in this work, the reward is influenced by factors such as the pedestrian's neighbors, static obstacles and the intended goal position. Due to the complexity caused by these influences, hand crafting the reward function to capture different desired behaviors is very difficult. The main goal of this work is to learn the reward function that best captures the pedestrian decision making process when navigating in a crowd. Next, we formally define our pedestrian modeling problem as an IRL problem.

C. Problem Statement

We define the pedestrian motion modeling problem in this section. Given an MDP with an unknown reward function R^*, formally denoted as MDP\R, a set of expert demonstrations $D = \{\zeta_1, \zeta_2, \ldots, \zeta_N\}$ generated by an agent following the demonstration policy π_D, and corresponding feature vectors ϕ for each demonstration, find the unknown reward function R^*, which can closely replicate the behaviors exhibited in D.

III. Deep IRL Approach

In this section, we present our deep IRL solution to the problem formalized above. We first present the neural network structure used to approximate the reward function and then provide the feature vector input we use in this work.

A. Maximum Entropy Deep Inverse Reinforcement Learning (MEDIRL)

The problem stated in Section II-C can be solved using the IRL framework. The main aim of IRL is to learn R^* that

drives the agent's action preferences. Specifically, it captures the reward structure that underlies specific agent behaviors. In this problem, IRL faces two main challenges namely suboptimal demonstrations, as no pedestrian will always act optimally and the ambiguity in the reward function itself, as numerous rewards can explain the same behavior. MEDIRL addresses these problems by modeling the demonstration behavior as a probability distribution over the demonstrated trajectories [17], [18] and then constrain it to the highest entropy [19]. MEDIRL also extends IRL to include non-linear reward functions. Due to the complex pedestrian decision making process in crowded environments, we assume that the reward function is a non-linear function of the feature vector $\phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$. The reward function R^* can be calculated using a Deep Neural Network (DNN) and can be defined as,

$$R^* = g(\phi, \theta_1, \theta_2, \theta_3, \ldots, \theta_j), \tag{1}$$

$$= g_1(g_2(\ldots(g_j(\phi, \theta_j), \ldots), \theta_2), \theta_1), \tag{2}$$

where θ=[θ_1,θ_2,...,θ_j] are the DNN weights, and $g_j()$ are nonlinear functions. DNNs are used due to their ability to represent highly nonlinear functions through the composition and reuse of the results of many nonlinearities in the layered structure and are regarded as *universal approximator* [17], [20]. The DNN structure used in this work is shown in Fig. 2. It consists of one input layer, two fully connected hidden layers and one output layer. The input to this structure is the feature vector ϕ and the output is the reward value R^*.

The problem of training the DNN can be framed in the context of Bayesian inference as MAP estimation, maximizing the joint posterior distribution of observing expert demonstrations D and the DNN parameters θ,

$$\mathcal{L}(\theta) = \log P(D, \theta | R^*) = \underbrace{\log P(D|R^*)}_{\mathcal{L}_D} + \underbrace{\log P(\theta)}_{\mathcal{L}_\theta}, \tag{3}$$

which allows this joint log likelihood to be differentiated with respect to the parameters of the neural network, θ, making it possible to apply gradient descent to this problem [21]. The objective function is given in the data term $\mathcal{L}_D$ of (3) and differentiable with respect to the reward function R^*, enabling back-propagation of the objective gradients to the neural network parameters θ. The final gradient is given as the sum of the gradient of the data term $\mathcal{L}_D$ and the model term $\mathcal{L}_\theta$ with respect to the network parameters θ

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}_D}{\partial \theta} + \frac{\partial \mathcal{L}_\theta}{\partial \theta}. \tag{4}$$

The gradient of the data term $\mathcal{L}_D$ can then be written in terms of the derivative of the expert demonstration with respect to reward R^* and the derivative of the reward function with respect to θ

$$\frac{\partial \mathcal{L}_D}{\partial \theta} = \frac{\partial \mathcal{L}_D}{\partial R^*} . \frac{\partial R^*}{\partial \theta}, \tag{5}$$

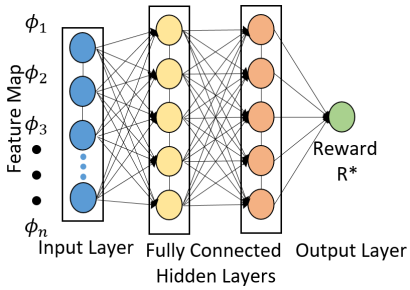


Fig. 2: DNN for reward function approximation based on the feature space represented by $\phi = [\phi_1, \phi_2, \phi_3, ..., \phi_n]$.

$$= \underbrace{(\mu_D - \mathbb{E}[\mu])}_{\text{State Visitation Matching}} \cdot \underbrace{\frac{\partial}{\partial \theta} g(\phi, \theta)}_{\text{Back-propagation}}, \tag{6}$$

where $R^* = g(\phi, \theta)$. The gradient of $\mathcal{L}_D$ with respect to R^* is equal to the difference in the exhibited state visitation frequency μ_D by D and the expected visitation counts $\mathbb{E}[\mu]$ for the learned systems trajectory distribution, which depends on the reward approximation given the corresponding optimal policy [19].

Algorithm 1 Maximum Entropy Deep Inverse Reinforcement Learning (MEDIRL)

Input μ_D^a,ϕ,S,A,T,γ
Output Optimal weights θ^*

θ = initialize_weights()
for m = 1:M **do**
  R^{*m} = g(ϕ,θ^m)
  MDP solution with current reward function
  π^m = approx_value_iteration(R^m,S,A,T,γ)
  $\mathbb{E}[\mu^m]$ = propagate_policy(π^m,S,A,T)
  Maximum entropy loss calculation and gradients
  $\mathcal{L}_D^m$=log(π^m)$\times\mu_D^a$
  $\frac{\partial \mathcal{L}_D}{\partial R^{*m}}$ = μ_D-$\mathbb{E}[\mu^m]$
  Compute network gradients
  $\frac{\partial \mathcal{L}_D^m}{\partial \theta_D^m}$ = nn_backprop(ϕ, θ^m, $\frac{\partial \mathcal{L}_D}{\partial R^{*m}}$)
  θ^{m+1} = update_weights(θ^m,$\frac{\partial \mathcal{L}_D^m}{\partial \theta_D^m}$)
end for

The overall MEDIRL framework is detailed in Algorithm 1. The algorithm assumes access to a set of expert training samples which are used to calculate μ_D. The feature vector ϕ for each training sample is also available. It also assumes the possible state values S and action value A are known. The state transition probabilities denoted by T, are assumed to be one for our case as we consider the pedestrian motion agent to be a deterministic system. The discount factor γ is a positive constant between 0 and 1, and is selected empirically.

The training procedure begins with randomly initializing the neural network weights θ. The initial network weights θ are then used to generate reward values R^* using the feature vectors ϕ for each training episode. The reward values R^* are then used to generate policies π for each

training episode of the current iteration m. The policies π are then used to calculate the expected state visitation frequencies $\mathbb{E}[\mu]$ for each training episode. The difference in the visitation frequencies from the policies π and the demonstrations are used to calculate the gradient $\frac{\partial \mathcal{L}_D}{\partial R^*}$, used to update the network weights θ by backpropogation. This process is repeated for M iterations. In the next section we present the features used for reward function calculation for this problem.

B. Feature Vector

In this section we present details of the features used as the input to the proposed MEDIRL algorithm. We define the state of the pedestrian i to be its spatial location $s_i = [x, y]$. The feature vectors can be generated using the spatial location of the pedestrian. The spatial location and velocity of its neighbors is also required. The features then extracted from these known values (from demonstration trajectories) include the following.

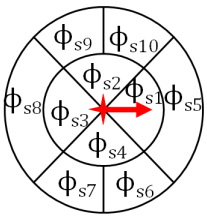


Fig. 3: The SAM feature represented by spatial location bin disks around the pedestrian of interest (i.e., the center). The pedestrian location is shown by the red cross and its motion direction is shown by the red arrow.

1) Social Affinity Map Feature (SAM): There is adequate evidence showing that people in large crowds are bound together by *social affinity*. Social affinity, which is defined as motion affinity of neighboring individuals [22]. Since pedestrians' movement is influenced by others in their neighborhood, the SAM feature clusters the spatial position of the pedestrian's neighbors in spatial bins [22]. The area around the pedestrian under consideration (the center), is divided into two concentric circles of radii r (e.g., 5 m) and r' (e.g., 3 m), as visualized in Fig. 3. The inner circle is subdivided into four equivalent bins. The outer ring between the two circles is first subdivided into four equal sections. Then the bins that are directly left and right of the pedestrian are further subdivided into two equal sections each. The pedestrian motion direction is shown by the red arrow. The presence of a pedestrian in each spatial bin is captured using a binary feature vector consisting of ten elements corresponding to the ten bins. This ten value feature vector is represented as ϕ_S = [ϕ_{S1}, ϕ_{S2},. . .,ϕ_{S10}]. The binary value of the corresponding element in the feature vector is set to one if there is a pedestrian present in that bin and vice versa.

We further extend the SAM feature from its earlier application in [22], to capture the velocity of the pedestrian's neighbors. This feature vector for each bin is represented as ϕ_{SV} = [ϕ_{o1}, ϕ_{o2},ϕ_{o3},ϕ_{v1},ϕ_{v2},ϕ_{v3}]. For each bin, we calculate the average velocity (speed l and heading α) of all the neighbors in that bin and set the binary values of the six feature vector for each bin according to the thresholds listed in Table I. The complete feature vector for all ten bins is thus represented as ϕ_{SC} = [ϕ_{SV1}, ϕ_{SV2},. . .,ϕ_{SV10}].

TABLE I: SAM: Velocity feature thresholds.

Feature	Thresholds
ϕ_{o1}	$\alpha \in (-\frac{3\pi}{4}, \frac{3\pi}{4}]$
ϕ_{o2}	$\alpha \in [\frac{\pi}{4}, \frac{3\pi}{4}) \bigcup [-\frac{3\pi}{4}, -\frac{\pi}{4})$
ϕ_{o3}	$\alpha \in (-\frac{\pi}{4}, \frac{\pi}{4}]$
ϕ_{v1}	$l \in$ [0,0.5) m/s
ϕ_{v2}	$l \in$ [0.5,1.0) m/s
ϕ_{v3}	$l \in$ [1.0,∞) m/s

Combing the two sets of vectors ϕ_s and ϕ_{sc} discussed above, the complete SAM feature vector is given by ϕ_{SAM} = [ϕ_S,ϕ_{SC}].

2) Density Feature: The *density feature* ϕ_d represents the number of neighbors in all the SAM bins [12], [15]. This feature consists of a three binary element vector represented by ϕ_d = [ϕ_{d1}, ϕ_{d2}, ϕ_{d3}]. Each element of the vector corresponds to a certain range of neighbors within the SAM circle of radius r. If the number of neighbors falls within the range corresponding to an element, that element is set to one and the other two are set to zero. If there are two or less neighbors in the SAM bins, ϕ_{d1} is set to one, otherwise it is zero. If there are greater than two but less than five other neighbors, ϕ_{d2} is set to one, otherwise it is set to zero. If there are five or more, ϕ_{d3} is set to one otherwise it is set to zero.

3) Distance Feature: It captures the distance between the pedestrian's current location to its goal position. This feature is represented as ϕ_{dis} and ensures the pedestrian moves towards its goal position.

4) Default Cost Features: This feature serves to balance the density, social affinity and the distance features as used in [12], [15], [23], [24]. It is denoted by ϕ_{def} and always set to 1.

The complete feature vector is thus obtained by concatenating the four feature presented above thus written as

$$\phi = [\phi_d, \phi_{SAM}, \phi_{dis}, \phi_{def}].$$

In the next section, we present implementation details of the approach developed in this section and the evaluation procedure.

IV. Expert Datasets and Evaluation Experiments

In this section, we present the details of the expert dataset used, the implementation details of the proposed deep IRL approach, and the procedure to evaluate the performance of the generated reward function.

A. Expert Dataset

The expert dataset used in this work, is open pedestrian trajectory data[1] collected in the "ATC" business center in Osaka, Japan [16]. A map of this environment has been shown

[1]http://www.irc.atr.jp/crest2010_HRI/ATC_dataset/

in Fig. 4 where the white objects show the static obstacle. The open pedestrian dataset was developed in [16] using the tracking system that consists of 49 range sensors installed above human head height to cover an area of about 900 m^2. The complete dataset consists of tracking data collected for approximately 3.25 million pedestrians, over a period of 92 days. The overall length of the trajectory of all the pedestrians in the dataset is 128,692 kms. The measurement frequency of the recorded data is approximately 25 Hz.

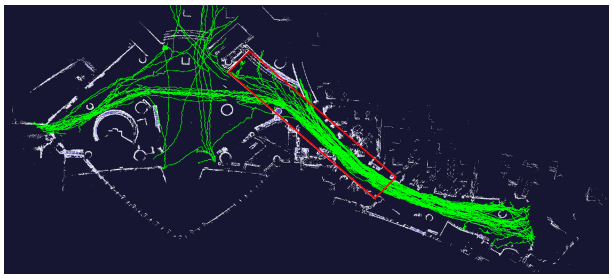


Fig. 4: The map of the mall with pedestrian trajectory data shown in green. Static obstacles are shown in white color. The red box denotes the area from which pedestrian trajectories were selected for learning R^*.

We processed the data using Python and OpenCV to visualize the trajectories on the map shown in Fig. 4. The green tracks mark the trajectories followed by the pedestrians in the environment. The trajectories in this figure show five minutes of pedestrian tracking data. Since the focus of this paper is to model pedestrian-pedestrian interactions, a smaller section of the tracking environment was selected, where the interaction of the pedestrian with static obstacles is minimal. That section is marked by the red rectangle in Fig. 4. Only pedestrians within this bounded region were used in the training process. The provided dataset includes a pedestrian ID, its spatial location, and its velocity at each measurement instance.

B. Implementation Details for Algorithm 1

The expert dataset was processed to extract training episodes for learning the reward function R^*. We designate a pedestrian's state at k-th time instance as S_k, feature vector as ϕ_k, the action vector a_k, that it executes to reach its future state S_{k+1} as a training episode designated by e_k. This vector can be formally written as e_k=[S_k,a_k,ϕ_k,S_{k+1}], and generated using the expert dataset. A series of these training episodes is referred to as a training epoch, which can be formally written as D_t=[e_1,e_2,. . .,e_W], where the total episodes are W = 10 million. Here it is important to note that the training episodes in D_t were not sequentially selected for the same pedestrian rather they were randomized after selection to reduce correlation in sequential training episodes and ensure generalization of the trained DNN [25].

The overall implementation of the dataset preprocessing, feature calculations and visualization are done using Python. The DNN and its training is implemented using TensorFlow. As a first step, the training episodes are extracted and stored. The DNN used in the implementation consists of one input layer, two hidden layers and one output layer. The two hidden layers consist of 4096 and 2048 fully connected nodes respectively. The environment around the pedestrian was discretized into a grid world following the implementation in [26]. Following the steps in Algorithm 1, the reward values are calculated using features corresponding to each training episode. These reward values are used in approximate value iteration to generate the optimal policy for the current step using line 4 in Algorithm 1. The state visitation error $\mu_D - \mathbb{E}[\mu]$ is back-propagated to update the DNN using gradient descent with learning rate set to 0.001. Typical MEDIRL problems are formulated for static environments which is not the case here. To tackle this problem, we re-plan after every time step of each episode. The solution is no longer optimal but it is considered to be a good approximation [15]. The training was performed on a dual NVIDIA GeForce GTX 1080Ti workstation with dual Xeon E5-2630v4 processors and takes approximately 20 hours.

Algorithm 2 Trajectory Evaluation

Input R^*,θ,S,A,T,ϕ,γ
Output Motion policy π

1: Load DNN parameters θ, that were trained by Algorithm 1
2: V_s=-∞
3: **repeat**
4: V_{t_s}=V_s;$V_{s_{goal}}$=0
5: $Q_{s,a} = R^*_{s,a}+\gamma \sum_{s' \in S} T_{s,a,s'} V_{s'}$
6: $V_s = softmax_a Q_{s,a}$
7: **until** $max_s(V_s - V_{t_s})<\epsilon$
8: $\pi^*(a|s) = e^{Q_{s,a}-V_s}$

C. Evaluation Experiments

Using the learned reward function R^* from the previous step, feature vectors, possibles states and actions, and discount factor, the pedestrian navigation problem is solved as an MDP sequential decision making process using approximate value iteration algorithm detailed in Algorithm 2. Here V_s is the optimal value for each state and Q is the Q-value.

In the next section, we present performance evaluation results of the proposed method.

V. Performance Evaluation

The performance of the learned reward function to capture pedestrian behavior (both in terms of fidelity to the generated path and exhibited behaviors) is evaluated in this section. We first present the accuracy metrics for the generated trajectories, and then present the learned behaviors.

A. Episode Performance Evaluation

The episode performance evaluation step is performed by checking the predicted action sequence using the learned R^* and value iteration in Algorithm 2 and calculating the state visitation error $\mu_D - \mathbb{E}[\mu]$ for a single episode. We extract 0.5 million episodes following the same method used for

the training step and calculate the correct action prediction accuracy to be 96.6 %.

B. Trajectory Performance Evaluation

The complete trajectory performance evaluation is performed for individual pedestrians using their initial position, final position, and the number of steps taken by the pedestrians to reach the final position. The *simulated trajectory* is the one generated by Algorithm 2 to evaluate the learned reward function R^* using the proposed MEDIRL approach, and the *measured trajectory* is the actual trajectory from the dataset. The complete trajectory is then compared with the measured trajectory of the pedestrian using the performance metrics detailed below.

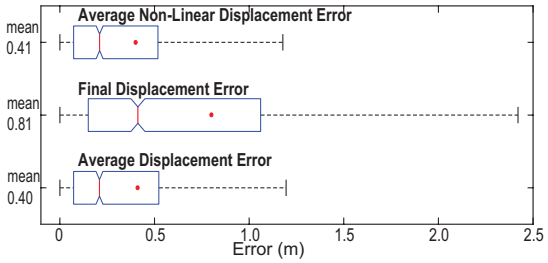


Fig. 5: Box plot of the errors between the simulated and measured pedestrian trajectories. The mean error value for each metric is shown by a red circle and the mean value is shown in the label along y-axis.

- *Average displacement error* is the mean square error (MSE) between the simulated trajectory and the measured trajectory [14], [27].
- *Final displacement error* is the distance between the pedestrian's measured final position and the simulated final position [14], [27].
- *Average non-linear displacement error* is the MSE in the non-linear sections of the pedestrian trajectory. Since pedestrians may turn when they interact with other pedestrians, this metric is important. We select the portions of the trajectory where the pedestrian changes its direction and check the error in those sections [14], [27].

These performance metrics were calculated for 1500 randomly selected pedestrian's trajectories from their initial to goal positions with their complete trajectories. The average length of trajectories is 16.3 m. The average duration of each trajectory is 16.03 secs. The simulated trajectories are then compared with the measured trajectories of the pedestrians. The displacement errors, final displacement errors and non-linear displacement errors are then calculated. The resulting error box plot is shown in Fig. 5. The figure also shows the mean error value with a red circle. The average displacement error is 0.40 m, the average final displacement error is 0.81 m and the average non-linear displacement error is 0.41 m.

A comparison of the results produced by our method with the results presented in [14] on these three metrics are detailed in Table. II. The randomly select 1500 pedestrians' trajectory are truncated to the duration of 4.8 secs, to be the same as the duration of predictions in [14] for fair comparison. The average length of trajectories for this case is 6.2 m. We generate these pedestrian's simulated trajectories and calculate the three errors described earlier and are listed in Table II.

TABLE II: Performance comparison.

Performance Metrics	Our algorithm	Algorithms in [14]	
	Average	Minimum	Average
Average Displacement Error	0.12m	0.09m(O-LSTM)	0.27m(Social-LSTM)
Final Displacement Error	0.27m	0.43m(IGP)	0.6m(SFM)
Average Non-Linear Displacement Error	0.11m	0.06m(O-LSTM)	0.15m(Social-LSTM)

The authors in [14] measure performance of several different algorithms with five datasets. We present the minimum and the average best case results across different datasets in Table II. The minimum average displacement error for any dataset is produced by the occupancy map-LSTM (O-LSTM) algorithm (0.09 m) and the minimum average error across all datasets is exhibited by Social-LSTM (0.27 m). The minimum final displacement error for any dataset is produced by the iterative Gaussian process (IGP) algorithm (0.43 m) while the minimum average error across all datasets is exhibited by the SFM (0.60 m). Finally the minimum average non-linear displacement error for any dataset is produced by the O-LSTM algorithm (0.06 m) while the minimum average non-linear displacement error across all datasets is exhibited by the Social-LSTM algorithm (0.15 m). It is important to emphasize here that since the performance in their case and our case is evaluated using different datasets, we cannot claim the superiority of one method over the other for any metric. The only aim of this comparison is to show that the performance of our method is numerically similar to other state of the art algorithms.

C. Individual Pedestrian Trajectory Analysis

In this section we analyze the complete trajectory of a pedestrian from its initial position to its goal position, designated as pedestrian C. Pedestrian C's measured trajectory is shown by the green line in Fig. 6. The initial position of the pedestrian C is marked with a yellow square and its goal position is marked with marked with a pink square. The trajectory generated using the learned reward function is shown in red. The neighboring pedestrians are marked with turquoise disks. The SAM feature circle is partially visible and shown in red and its center designates the location of pedestrian C. The static obstacles are shown in white color. There are certain interesting interactions of pedestrian C with its neighboring pedestrians, namely, leader follower behavior, and collision avoidance behavior, that we would like to discuss next. These interactions highlight the ability of our trained network to capture some well known behaviors exhibited by pedestrians.

1) Leader Follower Behavior: The leader-follower behavior refers to the case where a pedestrian follows another

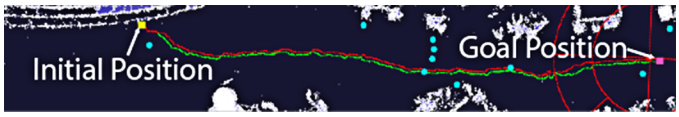


Fig. 6: The measured and simulated trajectory of the pedestrian C shown in green and red respectively. The initial position, goal position and SAM feature bins are also shown. Neighboring pedestrians are shown by turquoise disks.

pedestrian to mimic the behavior and adjust his/her motion in order to achieve smooth navigation [28]. The neighboring pedestrian thus becomes its leader. This behavior is exhibited when fast pedestrians pass slower ones in dense crowd situations. The follower behavior is exhibited by the pedestrian C. At t = 1.32 sec, another pedestrian appears in the tracking data marked in brown color in Fig. 7. The location of both pedestrians at every second are marked by red and brown disks respectively. Once the two approach the corridor at about 4 sec, they move closer to each other and form a leader-follower style formation. The pedestrian C assumes the role of the follower in this case. After t = 11 secs, the leader pedestrian speeds up, and moves away from the follower pedestrian. This separation of the two pedestrians highlights that they were navigating in a leader-follower formation under the influence of social affinity and not personal affinity, such as the affinity experienced by couples or people who know each other.

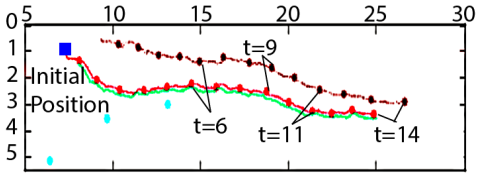


Fig. 7: Visualization of the leader-follower behavior that the pedestrian exhibits. The leader's measured trajectory, the follower's measured and simulated trajectories are shown in brown, green and red, respectively. The figure shows the *simulated* follower trajectory (in red) matches the *measured* follower trajectory (in green), thus validates our proposed algorithm.

2) Collision Avoidance Behavior: In this section, we analyze a section of pedestrian C's trajectory, where it avoids collision with oncoming pedestrians by subtly changing its trajectory. Fig. 8 shows an instance of pedestrian C on a collision trajectory with one of two oncoming pedestrian whose trajectory and location is shown in turquoise. The pedestrian C begins to move in a slightly upward direction, to avoid these two pedestrians at t = 7 secs. Once the two pedestrians pass, the pedestrian C moves downward again and continues along its original direction at t = 7.64 secs.

D. Group Behavior (Split-and-rejoin)

In this section, we present analysis of the motion of a group of pedestrians. For pedestrians walking as a group, for instance a *couple* [14], our proposed model performs well

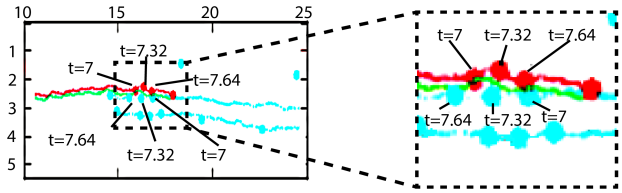
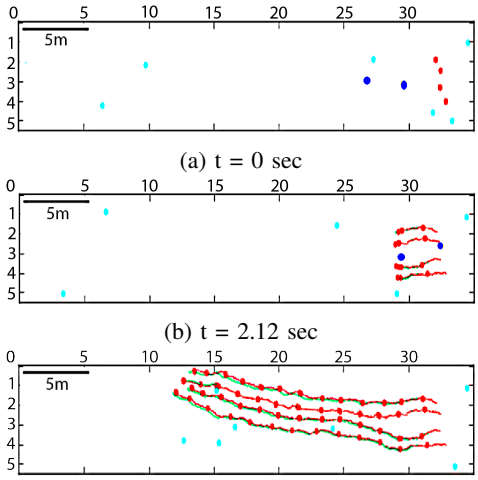


Fig. 8: Collision avoidance behavior of the measured (in green) and simulated (in red) pedestrian trajectories. The pedestrians execute maneuvers from 7 secs to 7.64 secs to avoid collision.



(a) t = 0 sec

(b) t = 2.12 sec

(c) t = 13 sec

Fig. 9: Group behavior (Split-and-rejoin) shown by a group of pedestrians. Two pedestrians passing between the group are shown in blue in Fig. 9a to Fig. 9b. The group is then shown to keep formation till 13 secs in Fig. 9c.

in predicting the motion of the individual pedestrians in the group. The behavior of such a group of pedestrians is shown in Fig. 9, where it shows the location of each pedestrian in the group with red disks, the simulated trajectory by red dotted lines, and measured trajectory of each pedestrian in green. Two pedestrians that interact with the group are shown by blue disks in Fig. 9a and Fig. 9b. The initial position of this group is shown in Fig. 9a. At the same time, two pedestrians moving in the opposite direction are also shown by blue disks. The group can be seen splitting in two subgroups in the middle, with two pedestrians moving upwards and two moving downwards to make room for the blue marked pedestrians moving in the opposite direction as shown in Fig. 9b. The two blue highlighted pedestrians as shown to have moved passed the group in Fig. 9b and the two subgroups move closer back to each other. The group then continues to move leftwards for the next 10.88 secs. The final location of the pedestrians are shown in Fig. 9c. It is important to highlight that our proposed model can capture complex group behaviors without explicitly modeling them. From the performance evaluation results presented above, we can see that our proposed deep IRL approach can successfully predict pedestrian navigation behaviors in

crowds.

VI. Conclusions and Future Work

In this paper, we present a deep IRL method to learn a reward function that captures the navigation behavior exhibited by pedestrians. We use open pedestrian trajectory data collected inside a mall as the expert dataset, which ensures that the learned reward function captures pedestrian motion behavior when navigating in a crowded environment. An MEDIRL algorithm was developed to learn the reward function from expert demonstrations, and feature vectors are selected utilizing SAM representing social affinity in human navigation behaviors. Evaluation procedures were developed to validate the proposed deep IRL method. We presented comparison of the simulated trajectories with the measured pedestrian trajectories in the dataset, which shows that the simulated pedestrian can replicate the measured trajectory with acceptable accuracy. Further analysis of our simulation comparison shows that natural human navigation behaviors can be replicated using our proposed method, which includes the individual pedestrian navigation behaviors such as collision avoidance, leader-follower and group behaviors such as split-and-rejoin.

In the future, we plan to extend this approach to use end-to-end IRL instead of using hand crafted features presented in this work. We also plan to incorporate pedestrian interactions with static obstacles that are not considered in the current paper. We will extend this method to learn the reward function for navigation by a robot in a pedestrian occupied environment. This learned reward function will be used to verify socially compliant navigation with robots and human subjects in field experiments. Using human pedestrian data in real-world would ensure seamless integration of robot navigating in human occupied environments.

References

[1] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, *et al.*, "MINERVA: A second-generation museum tour-guide robot," in *Proceedings of the International Conference on Robotics and Automation*, vol. 3, IEEE, 1999.

[2] H.-M. Gross, H. Boehme, C. Schroeter, S. Müller, A. König, E. Einhorn, C. Martin, M. Merten, and A. Bley, "TOOMAS: Interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 2005–2012, IEEE, 2009.

[3] H. Huttenrauch and K. S. Eklundh, "Fetch-and-carry with CERO: Observations from a long-term user study with a service robot," in *Proceedings of the International Workshop on Robot and Human Interactive Communication*, pp. 158–163, IEEE, 2002.

[4] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[5] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[6] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[7] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1688–1694, IEEE, Nov 2013.

[8] G. Ferrer and A. Sanfeliu, "Behavior estimation for a complete framework for human motion prediction in crowded environments," in *Proceedings of the International Conference on Robotics and Automation*, pp. 5940–5945, IEEE, May 2014.

[9] M. Hanheide, A. Peters, and N. Bellotto, "Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus," in *Proceedings of the International Symposium on Robot and Human Interactive Communication*, pp. 689–694, IEEE, Sept 2012.

[10] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine*, vol. 4, pp. 23–33, Mar 1997.

[11] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 797–803, IEEE, Oct 2010.

[12] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[13] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1343–1350, IEEE, Sept 2017.

[14] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 961–971, IEEE, June 2016.

[15] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1341–1346, IEEE, 2014.

[16] D. Brscic, T. Kanda, T. Ikeda, and T. Miyashita, "Person tracking in large public spaces using 3-D range sensors," in *Transactions on Human-Machine Systems*, vol. 43, pp. 522–534, IEEE, 2013.

[17] 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*, vol. 36, no. 10, pp. 1073–1087, 2017.

[18] M. Wulfmeier, P. Ondruska, and I. Posner, "Deep inverse reinforcement learning," *CoRR*, vol. abs/1507.04888, 2015.

[19] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Conference of Association for the Advancement of Artificial Intelligence*, vol. 8, pp. 1433–1438, AAAI, 2008.

[20] Y. Bengio, Y. LeCun, *et al.*, "Scaling learning algorithms towards AI," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[21] J. Snyman, *Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms*, vol. 97. Springer Science & Business Media, 2005.

[22] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 2211–2218, IEEE, June 2014.

[23] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 1083–1090, IEEE, Sept 2008.

[24] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Proceedings of the International Conference on Robotics and Automation*, pp. 981–986, IEEE, 2010.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] Matthew Alger, "Inverse reinforcement learning." `https://doi.org/10.5281/zenodo.555999`, 2017. Online; checked on June 12, 2017.

[27] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proceedings of the International Conference on Computer Vision*, pp. 261–268, IEEE, Sept 2009.

[28] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PLOS ONE*, vol. 5, pp. 1–7, April 2010.