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

Learning Human Navigation Behavior Using Measured Human Trajectories in Crowded Spaces

Muhammad Fahad[1], Guang Yang[2], and Yi Guo[2]

***Abstract*— As humans and mobile robots increasingly co-exist in public spaces, their close proximity demands that robots navigate following navigation strategies similar to those exhibited by humans. This could be achieved by learning directly from human demonstration trajectories in a machine learning framework. In this paper, we present a method to learn human navigation behaviors using an imitation learning approach based on generative adversarial imitation learning (GAIL), which has the ability of directly extracting navigation policy. Specifically, we use a large open human trajectory dataset that was experimentally collected in a crowded public space. We then recreate these human trajectories in a 3D robotic simulator, and generate demonstration data using a LIDAR sensor onboard a robot with the robot following the measured human trajectories. We then propose a GAIL based algorithm, which uses occupancy maps generated using LIDAR data as the input, and outputs the navigation policy for robot navigation. Simulation experiments are conducted, and performance evaluation shows that the learned navigation policy generates trajectories qualitatively and quantitatively similar to human trajectories. Compared with existing works using analytical models (such as social force model) to generate human demonstration trajectories, our method learns directly from intrinsic human trajectories, thus exhibits more human-like navigation behaviors.**

I. Introduction

Artificial intelligence has received booming research interest in recent years. Solving problems that are easy for people to perform but difficult to describe formally is one of the main challenges for artificial intelligence, such as recognizing spoken words or faces in images [1]. The human navigation problem falls directly in this category, where it is hard to define a universal set of rules to navigate in an environment with other humans and static obstacles. Robots will become part of the social fabric with increasing applications in our daily lives, for example, they are used in public spaces such as museums [2], supermarkets [3], and offices [4], and co-occupy the environment with humans. When navigating in human-centered environments, robots need to comprehend and comply with rules that humans follow, more than just avoiding collisions with other humans and obstacles, such as maintaining a comfortable distance from other humans, avoiding collisions, and maintaining coherency with the crowd [5]. Another challenge faced by existing robot navigation algorithms is failure to produce admissible trajectories, resulting in problems such as "robot freezing" [6]. Humans on the other hand are experts at generating admissible trajectories in even the most challenging conditions. Thus the best approach to navigation could be learning navigation from humans and then using it on a robot for conforming to desired characteristics such as comfort, naturalness and sociability [7] in addition to the ability of generating admissible trajectories.

Deep reinforcement learning (DRL) based approaches have received increasing research attention to solve robot navigation problems. Several studies (e.g. Chen et. al. [8], Long et. al. [9] and Everret et. al. [10]), investigated DRL approaches to develop control policies in the presence of other static and dynamic obstacles. The aforementioned DRL methods rely on handcrafted reward functions, which can only capture a finite number of rules, so only these rules are ingrained in the navigation policy.

A natural way to overcome the limitations of handcrafted reward functions for reinforcement learning is to learn directly from human examples to capture human-like behaviors. A maximum entropy IRL approach was used by Henrik et. al. to predict human trajectories in a controlled environment for navigation behavior between fixed points [5]. Alahi et. al. formulated the human navigation as a pattern recognition problem and adopted a Long Short Term Memory (LSTM) model to predict the sequence of actions, based on the past positions of the humans and their surrounding [11]. In our earlier work [12], we proposed to learn the reward function using IRL, and approximated the reward as a non-linear function of the input features including Social Affinity Map, density, distance and default cost features. This method required pre-processing of sensor information to extract meaningful features, before it could be used by the agent for navigation. Tai et. al. [13] presented a Generative Adversarial Imitation Learning (GAIL) based method to learn socially compliant navigation using simulations in a $Gazebo$ based environment, where the Social Force Model (SFM) [14] was used to simulate the trajectory of humans. Although SFM has been commonly used for simulations of human crowds in existing studies, it is parameter sensitive, and could limit the learned behaviors to only those exhibited by the SFM. Human behaviors on the contrary are much more diverse, learning from which benefits robot navigation, which is the main goal of this work.

In this paper, we focus on learning how humans navigate in crowds, and propose a GAIL based method to learn human

The work was partially supported by the US National Science Foundation under Grant CMMI-1825709.

[1] Muhammad Fahad, is with the Department of Robotics and New Product Development, National Oilwell Varco, Houston, TX, USA 77042 Email: muhammad.fahad@nov.com. The work was done during Muhammad's Ph.D. study at Stevens.

[2]Guang Yang and Yi Guo are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA 07030 Emails:{gyang11,yguo1} @stevens.edu.

navigation behaviors. The use of GAIL forgoes the reward function learning step required in IRL formulations, instead, it directly learns the control policy from demonstration data. We use a large open human trajectory dataset collected in a real-world business center [15], integrate it with a 3D robot simulator (V-Rep), and collect a training dataset, containing raw robot exteroceptive sensor measurements, that are obtained using the robot following the same trajectory as that followed by the human. We adopt learning from demonstrations approach, since human navigation behaviors draw from many complex experiences, thus it is challenging to manually craft a reward function for RL that captures all the behaviors exhibited by humans. Using real human trajectories, rather than those generated from analytical methods such as social force model (SFM) [14], the trained policy learns from human experiences rather than existing analytical models. The learned policy uses occupancy maps generated from a LIDAR sensor onboard the robot as input, and generates robot motion control directly. We compare the performance of the learned policy with the human demonstration trajectory and find them to be qualitatively and quantitatively similar.

II. Problem Formulation

In this section, we first present the robot motion model considered in this work. Then we present the formulation of the navigation problem in the Markov Decision Process (MDP) framework, followed by the formal problem statement.

A. Robot Motion Model

Intuitively considering, the motion of a human from its known initial position to a known desired destination can be obtained by integrating its velocity in the discretized time domain. At each time step, the human makes a decision to change its velocity, taking into account other humans, static and dynamic obstacles in the environment. We consider this decision of velocity change as the action in a Markov Decision Process (MDP). The human uses an unknown policy π_E, to make these decisions. Similar to the motion of a human, a robot's motion between its initial location and desired final position is controlled by a 2-D vector, (v, ϕ), that encodes the robot's linear velocity and heading respectively. Thus the complete trajectory of the robot from its starting position to its destination is obtained by integrating the robot's velocity over time. This robot motion has been illustrated in Fig. 1, where at each time step, the robot changes its velocity and heading and moves to a new position, at which, it performs the same process to continue its motion. In this work, we aim to learn a robot motion planning policy π_θ, that allows the robot to move in a crowded space in a way that mimics the unknown human motion planning policy π_E. The robot has been modeled as an agent whose decision making process, while navigating in a crowded space, follows an MDP.

B. Markov Decision Process (MDP)

In this section, we formalize the robot motion model in the MDP framework. An MDP is a probabilistic sequential

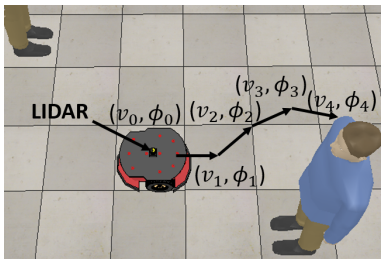

Fig. 1: Robot motion model as a sequence of headings and speed measurements. The complete trajectory of the robot is obtained by integrating its velocity over time.

decision making process where each decision, more formally an action, depends only on the agent's current state and is thus memoryless. The robot motion in a finite state MDP, with discrete time steps t = {1,2,...,T}, can be defined as a tuple (S, o_t, A, T, γ, R). The state space of the robot, S, at any time instance consists of the robot's current location **x**=(x_r, y_r), the linear velocity and heading (v, ϕ), and the desired heading to the destination ϕ_d at that time instance. The environment observation o_t at any time t is a 2-D occupancy map obtained from the point cloud measured using a LIDAR mounted on the robot. The occupancy map from the current time instance t and previous 3 time instances, t-1, t-2, and t-3, are stacked together to form o_t for the current time step. More details about the occupancy map are presented in Section IV-B. The action space A is the robot's linear acceleration and angular velocities (v', ω), which are obtained from expert trajectories during training. The state transition probabilities are denoted by T, while γ denotes the discount factor and R denotes the reward function. Each demonstration trajectory ζ is obtained as a time series of tuples (o_t, **x**, (v, ϕ), (v', ω), ϕ_d). The complete set of expert demonstration trajectory D_t is thus defined as $D_t = \{\zeta_1, \zeta_2, \ldots, \zeta_N\}$.

C. Problem Statement

We formally state our learning problem as: Given a set of expert demonstrations $D_t = \{\zeta_1, \zeta_2, \ldots, \zeta_N\}$ generated by measured human trajectories, find the policy π_θ, which can closely replicate the demonstration trajectories in D_t.

III. Proposed Method

This section presents the proposed approach for learning the robot motion planning policy π_θ, using the expert training samples generated by the robot following the human demonstration trajectory where the human motion is controlled by the unknown expert policy π_E. This can be cast as an imitation learning problem. The imitation learning framework used in this work is based on GAIL [16], which is an extension of generative adversarial networks (GAN) [17]. In contrast to Inverse Reinforcement Learning (IRL) used in our previous work [12], it forgoes the step of explicitly learning the reward function, but instead extracts the policy directly from the expert demonstrations. It trains a generator network G that learns the policy π_θ, parametrized by network

parameters θ, which in turn tries to confound the discriminator network D, parametrized by network parameters w. The goal of D is to distinguish between the data distribution of the observations and actions generated by the policy π_θ, with the data distribution of the expert demonstrations D_t generated by the unknown expert human motion policy π_E. The discriminator in GAIL framework is optimized using the loss function in (1), where $\lambda H(\pi_\theta)$ represents the causal entropy of the policy,

$$\mathbb{E}_{\pi_\theta}[\log(D_w(s,a))]+\mathbb{E}_{\pi_E}[\log(1-D_w(s,a))]-\lambda H(\pi_\theta). \quad (1)$$

The policy π_θ, is trained using a pseudo-reward function $\tilde{r}$ generated from the discriminator defined as

$$\tilde{r}(s,a;w+1) = -\log(D_{w+1}(s,a)) \quad (2)$$

which approaches infinity, as observations, state and actions sampled using π_θ become indistinguishable from D_t. The policy π_θ in this work is updated using PPO, which is a policy gradient method. PPO trains two networks that share parameters, namely an actor and a critic, where the actor is the generator in GAIL framework. Due to space limitation, we will not go into the inner workings of PPO and refer the reader to [18]. The overview of the training procedure

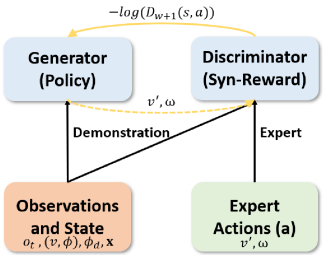

Fig. 2: The overview of training the policy network using the proposed learning framework. The discriminator learns from the expert dataset and generates synthetic rewards for updating the policy network using Proximal Policy Optimization (PPO).

in this proposed framework is shown in Fig. 2. At each update iteration, the agent interacts with the environment using the current policy π_θ to generate observations, states and corresponding actions. The generator training mini-batch τ_G is obtained from these interactions. The discriminator expert mini-batch τ_D is randomly sampled from D_t. The discriminator weights w are updated to $(w+1)$ to optimize loss function in (1) using Adam optimizer. The updated discriminator network is used to generated $\tilde{r}(s,a;w+1)$. The generator weights θ are updated using PPO with τ_G and $\tilde{r}(s,a;w+1)$. Note here that the environment is only used to generate the environment observations as a result of actions taken by the actor network and not the reward value. Also $\tilde{r}(s,a;w)$ is different from the reward under which the expert demonstrations were generated, but it can be used to drive π_θ into regions of the state-action space similar to that of π_E.

IV. Expert Dataset Generation

In this section, we present the details of the dataset used as expert dataset, and the method used to generate D_t for training and validation of the learned policy π_θ.

A. Human Trajectory Dataset

In Section III, we presented a GAIL based approach to train the policy π_θ, to enable a robot to navigate in the way a human would in crowded spaces. Any imitation learning method requires a training dataset, that consists of samples obtained when the agent we want to mimic, is performing the task that we aim to learn to perform. For the problem studied in this paper, the task is a human navigating in the presence of other humans, static and dynamic obstacles. Thus a human trajectory dataset is required to enable the method proposed in the previous section. The human trajectory dataset used in this work, is an open human trajectory dataset[1] collected in the "ATC" business center in Osaka, Japan [15]. The provided dataset is a time series that includes a unique ID used to identify the human during its tracked lifetime, its spatial location, and its velocity at each measurement time instance. The measurement frequency of the recorded data is approximately 25 Hz. The environment in which the human is tracked is shown in Fig. 3a. A total of 49 range sensors were installed above human height to cover this area of approximately 900 m^2. The complete dataset consists of tracking data collected for approximately 3.25 million humans, over a period of 92 days, with combined track length of 128,692 kms. We assume that each tracked agent in this database was planning its motion using the unknown policy π_E.

B. Synthetic 3D Data Generation

The original dataset is in the form of a time series, with spatial location, velocity and acceleration of each human in the environment, measured at each time instance. However, our formulation requires the state space, the action and most importantly the occupancy map o_t at each time step. The occupancy map o_t is created using robot exteroceptive sensors such as LIDAR. This training data could be collected by manually driving the robot in a human occupied environment and collecting the state space, the action and observations o_t. In that case, collecting such a large dataset would require a lot of effort and the collected data would be affected in two ways. Firstly, trajectories of the humans in the environment would be affected by the presence of the robot, since humans would behave differently in the presence of a robot than they would in the presence of a human [19]. Secondly, the trajectories thus generated would be from the third person perspective rather than the first person perspective, since the human controlling the robot would have a third person view of the robot's environment.

Our approach to overcome these challenges is to convert the 2D environment shown in Fig. 3a to a 3D environment and use simulated humans that follow the measured human

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

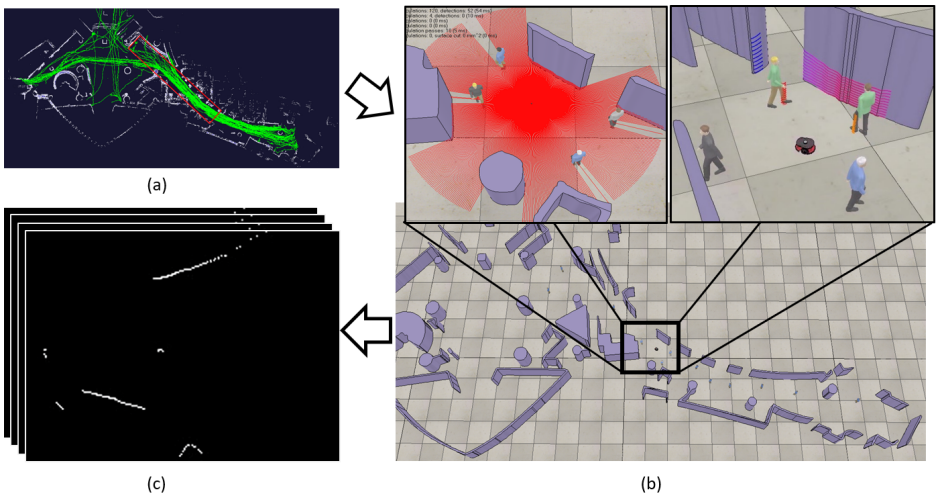

Fig. 3: Overall process used for creating the environment for expert demonstration generation and policy evaluation. The original 2D environment is shown in (a) along-with five minute persistent trajectory tracklets in green color. The region marked with red rectangle is the area inside which the trajectories were selected for expert training. The 3D V-rep environment with human and robot models, and LIDAR measurement field are shown in (b). The occupancy map generated from the LIDAR are shown in (c).

trajectories in the datasets to collect the expert data. The scaled 2D image file shown in Fig. 3a, was first extended in the z direction to a fixed height, and saved as an STL Mesh file. The mesh file was imported into V-Rep[2], a 3D robotics simulation software, is shown in Fig. 3b. The humans in the environment were simulated using built-in V-Rep human models, which were controlled to follow the measured trajectories of the humans in the dataset in this 3D environment. A LIDAR equipped Pioneer 3DX chassis was added in the simulator to produce o_t.

Each tracked human for which training episode ζ was to be obtained, was selectively replaced by the robot model. The robot was then controlled to follow the measured trajectory of this human. The remaining human models in the environment were kept the same and were controlled to follow their real trajectories in the dataset. The observations o_t, robot state (v,ϕ), ϕ_d, $\mathbf{x}$, and actions (v',ω) were then collected and stored for training.

The observations o_t were stored as occupancy maps, created by converting the point cloud from the LIDAR, to grid locations plotted on a scaled monochromatic image shown in Fig. 3c. Here it is worth mentioning that the complete LIDAR scan was not used, instead the 360 degree scan, along the horizontal direction was used. Using just the horizontal scan ensures that our method is not affected by factors such as orientation of the neighboring human relative to the robot as long as at least one beam of the LIDAR is reflected by the neighboring human.

V. Policy Training and Evaluation Method

In this section, we present the procedure used to train both the policy and the discriminator network, and the procedure to infer trajectories using the trained policy network.

[2]http://www.coppeliarobotics.com/

A. Policy Training Algorithm

As discussed in Section III, the proposed problem solution requires two parametrized models, namely, the actor/critic (generator) and discriminator. The critic network, referred to as the value network, is required to learn the policy π_θ using PPO, but neither this network, nor the discriminator network are used once training is complete. The parameters of these models have been designated by θ and w respectively. These two models have been implemented using deep feed forward networks consisting of Convolutional Neural Network (CNN) layers, followed by fully connected layers. The CNN layers extract the necessary features for both the action/critic network and the discriminator network. Dense fully connected layers 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* [20], [21].

As shown in Fig. 4, both deep neural networks consist of three CNN layers, with kernel sizes, 5, 5 and 3 respectively, stride 2 and Rectifier Linear Units (ReLU) activation functions. These are followed by fully connected layers with 512 hidden units each, which use ReLU activation functions, with the main difference in the output layers. The output layers of actor and critic networks have linear activation functions, while the discriminator uses Sigmoid activation function. The complete training procedure is listed in Algorithm 1. The weights θ and w are randomly initialized. For each training episode the expert training mini-batch τ_D is sampled from D_t. The current policy π_θ is used by the robot to interact with the environment to obtain generator samples. A subset of these are selected as τ_G. The discriminator weights w are updated to $w+1$ according to (1) using Adam Optimizer. The updated discriminator D_{w+1} is used to generate syn-

Algorithm 1 Algorithm For Policy Network Training

Input Demonstration trajectories D_t
Output Policy π_θ weights θ

Randomly initialize policy network π_θ weights θ
Randomly initialize discriminator network D_w weights w
for m = 1:M **do**
  for n = 1:N **do**
    Sample τ_D consisting of o_t, (v,ϕ), ϕ_d, **x**, and (v',ω) from demonstration trajectories D_t
    Sample τ_G consisting of o_t, (v,ϕ), ϕ_d, **x**, (v',ω), $\tilde{r}(s,a;w+1)$ and $V(s)$ after robot interaction with the environment with current version of π_θ
    Update the discriminator parameters from w_n to w_{n+1} according to (1) using Adam Optimizer
    Update policy parameters θ_n to θ_{n+1} using the PPO rule with synthetic reward $-log(D_{w+1}(s,a))$
  end for
end for

thetic rewards $-log(D_{w+1}(s,a))$ and update policy network weights θ_n to θ_{n+1} using the PPO. These interleaved Adam Optimizer and PPO iterations are performed for the N expert episodes and M epochs.

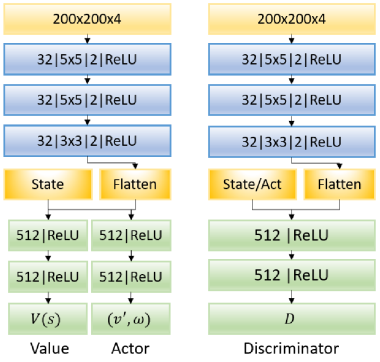

Fig. 4: The network architectures used for actor/critic and discriminator networks, where inputs are occupancy maps, current states and additionally actions for the discriminator.

B. Evaluation Experiments

The learned policy π_θ is used to generate evaluation trajectories by randomly selecting a human trajectory in the measurement dataset, not included in the training trajectories. The initial location of the robot is set to be the same as the initial location of the selected human. The motion of the robot model is controlled using π_θ. The positions of the remaining humans in the environment are updated according to their respective measured positions in the dataset at each time step. A new LIDAR scan is obtained with each time step and the position of the robot is updated. This is continued for the same duration as the duration of the human trajectory in the original dataset. The trajectory in the original dataset is referred to as the *measured trajectory* and the one inferred by the robot using π_θ is referred to as the *simulated trajectory*.

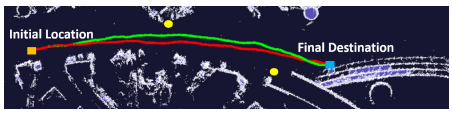

Fig. 5: Comparison of measured trajectory (green), of the human and simulated trajectory (red) of the robot generated using the trained control policy. Initial location and the final destination are also shown.

A sample trajectory thus generated is shown in Fig. 5, in which the robot navigates from its initial location to its final destination. The measured trajectory is marked with green color while the generated trajectory is shown in red. The initial and final locations are marked with orange and blue markers respectively. The location of other humans in the environment is shown by yellow disks, while the location of static obstacles is shown by white color. The same coloring convention is used for the remainder of this work. A total of 200 such trajectories were generated using π_θ for the performance evaluation discussed in the next section.

VI. Performance Evaluation

In this section, we present the results of the performance evaluation of the learned control policy both qualitatively and quantitatively, and compare these with current state of the art methods.

A. Quantitative Performance Evaluation

The measured trajectories of 200 randomly selected humans in the dataset, not part of D_t, were quantitatively compared to their respective simulated trajectories on three metrics, namely, *Average Displacement Error (ADE)*, *Average Non-Linear Displacement Error (ANLDE)* and *Final Displacement Error (FDE)*. These metrics have been previously used in similar studies [11], [12]. ADE and ANLDE are the Mean Square Error (MSE), for the complete and non-linear sections of the measured and simulated trajectories respectively. Non-linear sections are defined as time steps where ω is greater than 0.01 rads. FDE is the error between the simulated and measured final positions of the robot and the corresponding human.

TABLE I: Performance comparison of proposed algorithm with existing state of the art methods.

Performance Metrics	Our Method	IRL [12]	Algorithms in [11]	
	Avg	Avg	Min	Avg
Average Displacement Error	0.28m	0.12m	0.09m(O-LSTM)	0.27m(S-LSTM)
Final Displacement Error	0.58m	0.27m	0.43m(IGP)	0.6m(SFM)
Average Non-Linear Displacement Error	0.29m	0.11m	0.06m(O-LSTM)	0.15m(S-LSTM)

The comparison of these metrics with our earlier IRL based method [12], and other existing state of the art methods [11] is shown in Table. I. To keep a consistent comparison

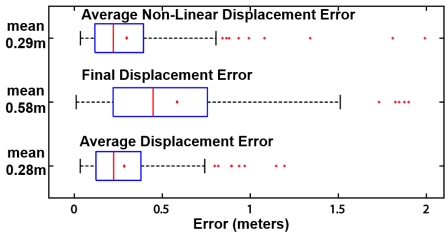

Fig. 6: Boxplot of the error metrics of the simulated trajectories and measured of humans. The trajectories are compared for average displacement error, final displacement error and average non-linear displacement error.

basis with those existing works, the maximum trajectory duration is truncated to 4.8 secs. The average length of trajectories is 6.3 meters. Our proposed method's average errors are quantitatively similar to the methods detailed in [11]. The error metrics are slightly higher compared to our IRL based approach which uses the same dataset, but the main difference in this work is the use of direct sensor measurements, rather than discretized, calculated features as in our IRL based work. The aim of this comparison is not to claim the superiority of one method over the other, since methods in [11] use different training datasets. The goal of this quantitative comparison is to show similar error metrics of our proposed approach with existing methods.

The boxplot of the 200 simulated human trajectories is shown in Fig. 6. The mean ADE is 0.29 m, the mean FDE is 0.58 m, while the mean ANLDE is 0.28 m. The box plot shows that the ADE of all the trajectories evaluated is within 0.8 meters of the measured pedestrian trajectory. These results also show that the robot controlled by the learned π_θ, reaches within 1 meters of the measured final position of the pedestrian 86% of the times, and within 1.5 meters 100% of the times. The ANLDE is less than 0.7 meters for 100 % of the trajectories. This is significant, since according to [11] (and also intuitively, since an agent only changes its heading from a straight line path to its destination to avoid collisions with other static and dynamic obstacles), the non-linear sections of trajectories mark the instances where the agent changes its heading, due to interaction with other humans and obstacles.

B. Individual Trajectory Analysis

In this section, we analyze certain qualitative behaviors exhibited by humans in measured trajectories, which are captured in the learned policy.

1) Collision Avoidance - Static Obstacles: A demonstration of the collision avoidance behavior with static obstacles, learned by the policy can be observed in Fig. 7. It can be seen that the robot model has learned to move around the pillar of the business center to avoid colliding with it in the same way a human would.

2) Collision Avoidance - Humans: The trained model exhibits collision avoidance behavior with other humans, as shown in Fig. 8. The simulated robot model's trajectory deviates slightly from the measured trajectory of the human.

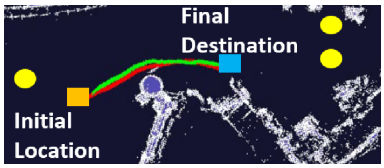

Fig. 7: The simulated robot model moves around a static obstacle to avoid colliding with it.

Despite this deviation, the simulated model starts to move upwards at 8.6 secs, to avoid collision with the oncoming human. At 9.28 secs, once the oncoming human has passed the robot, the robot starts to move downwards again towards it original position. This behavior shows that the simulated model took successful evasive action and was able to avoid collision with the oncoming human.

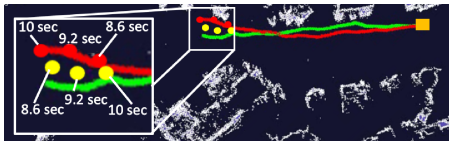

Fig. 8: Collision avoidance behavior exhibited by the trained model in response to an oncoming human. The simulated trajectory deviates from the measured trajectory, but the simulated model still moves upwards to avoid collision with the oncoming human.

In general, we analyzed collisions with both static and dynamic obstacles for all 200 trajectories and found only one instance of collision with a static obstacle. During the analysis, we also observed that at that instance, there was significant change in the measured trajectory of the pedestrian, likely due to noise in sensor measurement. Otherwise, the policy π_θ can generate collision free trajectories, even when quantitatively ADE, FDE and ANLDE values are high.

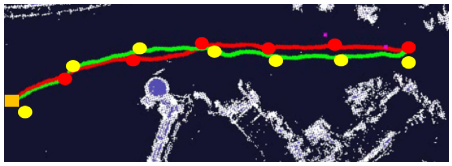

Fig. 9: The simulated robot model forms a leader follower formation with a neighboring human while moving towards its destination. The locations of simulated robot and neighboring human are marked at 4 sec intervals by red and yellow disks respectively.

3) Leader Follower Behavior: The ability of the learned policy to perform formation control in the form of leader follower behavior can be observed in Fig. 9 where the robot model controlled by π_θ follows a human and adjusts its motion to achieve smooth navigation as expected from a human. The measured trajectory of the human is shown in green color, while the simulated trajectory generated by the learned policy is shown in red color. The locations of the robot and the neighboring human that it forms the leader

follower formation with, are shown by red and yellow discs respectively, at 4 sec intervals. It can be seen that the learned policy is able to maintain a leader follower formation with the human ahead of it which is typically formed by humans even without knowing each while moving in close proximity [22].

C. Discussion

In this section, we validated the presented algorithm statistically, for quantitative metrics and qualitatively for different interaction scenarios. The primary goal of this paper is to develop and validate a GAIL-based algorithm capable of learning human navigation behaviors. The occupancy map used in the training phase can be generated using any physical LIDAR sensor, as long as the maximum range of the LIDAR is truncated to the maximum range used during training, and the scale of the occupancy map is kept the same as in the training phase. Thus the learned policy π_θ in simulation can be used and validated in real world experiments, which is part of our future work.

VII. Conclusion and Future Work

In this paper, we presented an algorithm based on GAIL to learn human navigation behaviors in a crowded space. We used an existing open human trajectory dataset collected inside a mall environment as expert samples. We developed a novel method to convert this dataset into a 3D environment and collected robot exteroceptive sensor data. An imitation learning based method is proposed to learn human navigation behaviors in the presence of other humans and static obstacles. The proposed method was evaluated for quantitative similarity of trajectories generated with the learned policy and the corresponding measured trajectories of the humans. The performance of the proposed algorithm in this work was shown to be quantitatively similar to existing works. We also evaluated the ability of the learned policy to capture behaviors of the humans such as collision avoidance with static obstacles and other humans and leader follower behaviors.

In the future, we plan to extend this work by using multiple training datasets to train the navigation policy and evaluate these trained policies without retraining in completely unforeseen datasets. In addition, we plan to conduct experiments with real robots and evaluate the performance of the trained policy in real world experiments. We also plan to make the 3D simulation environment and the collected datasets open for community use to advance research in robot social navigation.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville in *Deep Learning*, ch. 1, pp. 1–2, Cambridge, MA: The MIT Press, 2016.

[2] 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 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 1999.

[3] 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 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2005–2012, 2009.

[4] 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 IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, pp. 158–163, 2002.

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

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

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

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

[9] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6252–6259, May 2018.

[10] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, Oct 2018.

[11] 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 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, June 2016.

[12] M. Fahad, Z. Chen, and Y. Guo, "Learning how pedestrians navigate: A deep inverse reinforcement learning approach," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 819–826, Oct 2018.

[13] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1111–1117, 2018.

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

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

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

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] Z. Chen, C. Jiang, and Y. Guo, "Pedestrian-robot interaction experiments in an exit corridor," in *IEEE International Conference on Ubiquitous Robots (UR)*, pp. 29–34, June 2018.

[20] 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 (IJRR)*, vol. 36, no. 10, pp. 1073–1087, 2017.

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

[22] 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.