A Generative Approach for Socially Compliant Navigation

Chieh-En Tsai

CMU-RI-TR-19-28

July, 2019

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Supervisors:
Dr. Jean Oh
Dr. Luis Navarro-Serment

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

For my parents and my sister,
whose unwavering support make me a man dares to dream.

Abstract

Robots navigating in human crowds need to optimize their paths not only for the efficiency of their tasks performance but also for the compliance to social norms. One of the key challenges in this context is due to the lack of suitable metrics for evaluating and optimizing a socially compliant behavior. In this work, we propose NaviGAN, a generative navigation algorithm in an adversarial training framework that learns to generate a navigation path that is both optimized for achieving a goal and for complying with latent social rules. Different from the reinforcement learning approaches who only covers the *comfort* aspect of socially compliant navigation, and inverse reinforcement learning approaches who only covers the *naturalness* aspect, our approach jointly optimize both *comfort* and *naturalness* aspects. The proposed approach is highly interpretable and demonstrates superior quantitative performance in sets of experiments. We also demonstrates qualitative performance on a ClearPath Husky robot and perform extensive experiments on real-world robotic setting. The video of qualitative robot experiments can be found in the link: https://youtu.be/61blDymjCpw

Acknowledgments

I would like to thank Jean Oh for her mentorship over the past two years. I enjoyed every discussion I had with Jean. I would also like to thank Luis Navarror-Serment, Maggie Wigness, and John Rogers for their help on putting every pieces of our robot system together. Also Yeeho Song, Junjiao Tian and the BIG group for the time we spent together. Special thank to Yubo Zhang, thank you for always be by my side and make me the happiest man in the world.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will motivate the problem of "socially compliant navigation" and highlight the challenges that are involved. We will then introduce another challenging problem of "human trajectories prediction" that is closely related to social navigation. Finally, we describe the organization of the thesis at the end of this chapter.

1.1 Socially Compliant Navigation

In the context where robots co-exist and collaborate with people, social compliance becomes an important objective in designing robot behaviors. Social compliant navigation refers to the robot navigation problem where robots need to consider not only their own task-related objectives, but also social compliance objectives (e.g. delivering a mail to an office while keeping an appropriate distance from pedestrians). Toward this goal, a robot must have the ability to perceive and understand crowd behavior to plan its future trajectory accordingly.

There has been several publications describing prototype robots that operate in public spaces with humans. Museum tour-guide robots [6, 7, 11, 30, 32, 36, 41] are the main interest of socially compliant navigation research as the museum setting is particularly suitable for deployment of robot prototypes. Jensen et. al. [20] presents an overview of the design and implementation of a tour-guide robot (RoboX) where the challenges of reliability and safety in public space have been highlighted.

Those early attempts have navigation modules that prevent collisions with humans

Figure 1.1: An example illustrating the complexity of navigating in human crowds.

or other obstacles using strategies such as minimal distance to detected obstacles or stop-and-wait behavior in near-conflict situations. However, those algorithms who only focus on the next step is short-sighted. To plan a long-sighted path, *human trajectory prediction*, which we will introduce in the next section, becomes a focus among the researchers. Even with the ability to predict trajectories of other agents, in a crowded scenario, socially compliant navigation still faces the "Freezing Robot Problem" [42] as most of the researches dealt with persons separately and therefore ignore the interaction between social agents.

As identified in [26], there are three areas of research within the larger topic of socially compliant navigation where the acceptance of such robot agents can be improved:

- **Comfort** is the absence of annoyance and stress for humans in interaction with robots.
- **Naturalness** is the similarity between robots and humans in low level behavior patterns.
- **Sociability** is the adherence to explicit high-level cultural conventions.

They note that comfort is different from safety as even when a robot is moving safely, an observer may still feel that the motion is not safe as one lack trust in the

technology. Under this definition, researches on comfort area attempt not only to make the robot move safely, but also in a way that make it feels safe to humans.

Most of the reinforcement learning [21, 24, 38, 40] based methods [8, 9, 10, 13, 28] for socially compliant navigation are designed to optimize the comfort aspect. Chen et. al, [8, 9] proposes to use Deep V-learning in combination with hand crafted reward function that penalizes collisions and uncomfortable distances to learn the value network who induces optimal policy that maximizes the expected reward. The reward function to evaluate and optimize the social awareness of a robot policy used by [8, 9, 10] is defined as

$$R_t = \begin{cases} -0.25 & \text{, if } d_t \leq 0 \text{ meter} \\ -0.1 + \frac{d_t}{2} & \text{, else if } d_t < 0.2 \text{ meters} \\ 1 & \text{, else if reach goal} \\ 0 & \text{, otherwise} \end{cases} \tag{1.1}$$

Where R_t is the reward at time t and d_t denotes the minimum separation distance between the robot and the humans during execution of last action. Such reward function captures only the dynamic collision avoidance aspect which only cover the safety requirement, and probably mild comfort requirement, of socially compliant navigation. As socially compliant navigation requires richer model than only penalizing on collisions and uncomfortable distances, the social reward R_t is non-trivial to specify.

To cover the naturalness, another stream of research take the "learning from demonstration" approach. More specifically, [23, 25, 33, 35, 44] utilizes inverse reinforcement learning [1, 2, 31, 47] to jointly learn a reward function and a policy for socially compliant navigation. By jointly learning the reward and policy, they take a purely data-driven approach to uncover latent aspects of social navigation from expert demonstration. As the navigation policy is learned directly from human demonstration, it is able to capture the behavior patterns that humans possess and therefore cover the naturalness of socially compliant navigation. However, as the expert demonstrations tend not to contain failure cases (e.g., pedestrians do not collide during recording), the safety aspect of socially compliant navigation is only implicitly captured (i.e., model has to figure out the implicit safety aspect from demonstration without explicit clue of what to learn and what not to learn). This

makes inverse reinforcement learning based method hard to generalize when the number of pedestrians increases. Therefore, most of the previous works in inverse reinforcement learning for socially compliant navigation only learn from, and test on, a small number of pedestrians (e.g., 3). We will cover more details of algorithms for socially compliant navigation in Chapter 3.

1.2 Human Trajectories Prediction

In order to plan a long-term human-aware path that enables lots of robotics senarios (e.g., robot-guide, self-driving delivery service ...), human trajectories prediction problem recently gains more and more focus among researchers. Human trajectories prediction refers to the ability to predict the trajectory that each social agent will take given their previous trajectories as observation. In order to correctly predict the trajectory for each social agent, we need to model the interactive behavior among social agents including collaborative behavior. Existing works [4, 5, 18, 29] in human trajectories prediction focus on modeling the interactions between human pedestrians.

The Social-LSTM networks [3] attempt to model social behavior in a local scale by introducing the idea of social-pooling with LSTMs. The procedure of social-pooling constrains the space around a target pedestrian into fixed-size grids known as social grids, and then performs a pooling operation to integrate the information from the LSTM cell states of the neighboring pedestrians in the social grids. Social-LSTM, performs well, in general, in modeling human interactions for human trajectories prediction. However, the social-pooling assumes that only the neighbors within proximity affect pedestrian behaviors, which can be insufficient in dynamic environments (e.g., if a pedestrian from the outside of the social grids is running toward the target pedestrian, the target agent might react more to that running agent rather than attending only to the neighbors in the social grids).

In order to achieve a global view of all pedestrians during modeling, Social-Attention [45] represents the temporal relations and the spatial relations using a structured recurrent neural network (S-RNN) [19], and assigns an attention weight on each relation edge of the S-RNN. The attention mechanism allows the network to selectively attend to those pedestrians of interest while getting a fine-grained global view of all of the pedestrians in the scene.

All of the aforementioned methods for human trajectories prediction share the same objective of minimizing the negative likelihood of the ground truth trajectory. This objective assumes that the observed expert demonstrations are the only optimal solutions, which may not always be the case in the real-world crowd behavior where there can be multiple trajectories that achieve the same goal while also being socially compliant. Instead of using the negative likelihood as the sole objective, Social-GAN [15] and SA-GAIL [48] use generative adversarial networks (GANs) to train a generator to predict crowd behavior. Whereas these approaches focus on generating diverse trajectories conditioned on random noise, our approach focuses on performing socially compliant path planning conditioned on goals. In Chapter 2, we will cover more details of algorithms of human trajectories prediction that motivate our socially compliant navigation solution.

1.3 NaviGAN for Socially Compliant Navigation

The majority of existing approaches on social-aware path prediction focus on modeling local, reactive behavior of human pedestrians as opposed to planning for navigation among crowds. Toward the goal of developing a navigation algorithm that can produce socially compliant paths, we address several key challenges. First, the approaches derived from predicting human pedestrian trajectories focus on mimicking human demonstrations by using the L2 loss function. Such formulation makes it difficult to train general social navigation behavior because the objective assumes that the demonstrated actions are the only optimal choices when there can be alternatives. Next, although deep networks [3, 15, 45] are the winning approaches here, their lack of interpretability is a common issue as in other domains. Finally, the L2 metric for evaluation captures only the difference of generated path from ground truth recording. As the concept of socially compliance is complicated and the ground truth is not the only solution that is socially compliant, we seek more metrics for evaluating the performance.

To address these issues, we present NaviGAN–a generative approach for performing socially compliant robotic navigation that covers both *comfort* and *naturalness* aspects of socially compliant navigation. We formulate the problem as social navigation of a *target agent*– the agent that is performing navigation toward its goal state in a

populated environment. Aiming at improving both performance and interpretability, we harness the success of the social force model [16] to separately model the *intention* of a target agent and its *social interactions* with other agents. We propose NaviGAN as a computation model for representing social force. Our approach builds on the idea of an encoder-decoder [37] design of long short-term memory (LSTM) [17]. In the proposed model, we design a separate encoder-decoder LSTM to represent each different type of influence (or force) on navigation decisions that can be due to an intention, social interaction, or natural randomness of human behavior.

Motivated by increasingly popular generative adversarial networks (GANs) [14]based method, we pair the force generators with a social-aware discriminator that learns to identify socially compliant navigation paths. In comparison to reinforcement learning based methods [8, 9] that learns a policy to maximize the expectation of a hand-crafted reward function, NaviGAN takes a purely data driven approach to discover the latent social aspects encoded in human traveling data through adversarial training and generate a policy that is aware of those social aspects.

To evaluate the performance of NaviGAN, we carry out experiments using the same settings and metrics as the baselines. In addition, we perform a set of experiments that predict a long-term (24 sec) trajectory in crowd recording playback and measure not only the social score 1.1 of the algorithm, but also the *comfort rate* (percentage of episodes whose robot position is never getting closer than a comfort threshold to other pedestrians) and *arrival rate* (percentage of episodes whose robot ends up reaching the goal point). Our approach excel at achieving superior comfort rate while maintaining decent arrival rate. We deploy our algorithm on a mobile robot to perform real-world testing.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 presents in details the past works in the domain of human trajectories prediction in dense crowd that motivate our socially compliant navigation solution. Chapter 3 formally defines the problem of socially compliant navigation and survey most recent solutions to the problem which can be categorized to two classes: reinforcement learning approaches and learning from demonstration approaches. We will also compare and discuss the

advantages and limitations of those approaches. Chapter 4 presents our generative approach to solving socially compliant navigation which works in dense crowd and focus on both comfort and naturalness requirements of socially compliant navigation. Result of experiments and real-world mobile robot deployment are also presented. Finally, we conclude our work and discuss future directions of research in socially compliant navigation in Chapter 5.

Chapter 2

Literature Review - Human Trajectories Prediction

In this chapter, we present in details some past literature in the domain of human trajectories prediction in dense human crowds that motivate our socially compliant navigation solutions. These algorithms will provide background for discussion in later chapters and will serve as baselines of some experiments in Section 4.1.5.

2.1 Problem Definition

Let $\mathcal{D}$ denote a set of agents (pedestrians) who are involved in the time sequence $\{1, \ldots, T_{obs}, \ldots, T_{end}\}$. Given target agent i in $\mathcal{D}$, let $X_i = \{x_i^1, \ldots, x_i^{T_{obs}}\}$ be the past state sequence of agent i for time steps $t = 1 \ldots T_{obs}$; and $Y_i = \{y_i^{T_{obs}+1}, \ldots, y_i^{T_{end}}\}$, the demonstrated human trajectory provided by agent i, where x_i^t and $y_i^{t'}$ denote the state of agent i at time step t and t', respectively. For each agent i in $\mathcal{D}$, given a sequence of past trajectories of all agents $\{X_j : j \in \mathcal{D}\}$, we aim to generate a state sequence $\hat{Y}_i = \{\hat{y}_i^{T_{obs}+1}, \ldots, \hat{y}_i^{T_{end}}\}$ up to time step T_{end} with an objective of demonstrating human-like behavior.

2.2 Backgrounds

2.2.1 Social Force Model

Helbing and Molnar [16] proposed the social force model, summarized in Equation (2.1), for pedestrian behavior where the dynamics of the pedestrian $\vec{F}_{ped}$ is composed of an attractive force $\vec{F}_{intent}$ that model the intention of the pedestrian, two repulsive forces $\vec{F}_{social}$ and $\vec{F}_{obs}$ that model the interactions between the pedestrian with others and with static obstacles, respectively, and a fluctuation force $\vec{F}_{fluct}$ which is caused by the stochastic nature of pedestrian behavior.

$$\vec{F}_{ped} = \vec{F}_{intent} + (\vec{F}_{social} + \vec{F}_{fluct}) + \vec{F}_{obs} \tag{2.1}$$

The social force model achieved an early success in modeling human behavior but lost attention recently due, in part, to the surge of deep learning methods such as Social-LSTM and Social-Attention. When compared to such blackbox algorithms, the formulation of social force is more human interpretable, which can be of crucial importance to robot applications navigating among human crowds.

2.2.2 Adversarial training

A generative adversarial network (GAN) consists of a generator G and a discriminator D training in opposition to each other. While generator G is responsible for generating realistic samples $G(z)$ conditioned on some random noise z, discriminator D is responsible for estimating the probability of the sample being drawn from the real training data rather than generated by the generator G. The training procedure of an adversarial training is formulated as a min-max game with the following objective:

$$\min_G \max_D \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log D(x)] + \mathop{\mathbb{E}}_{z \sim p(z)} [\log(1 - D(G(z)))]. \tag{2.2}$$

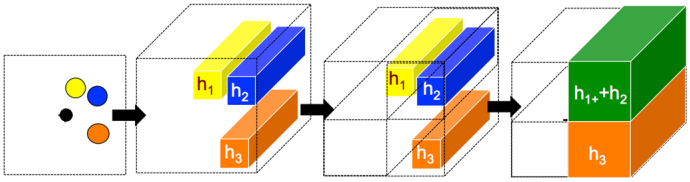

Figure 2.1: Social pooling assumes interactions only happen in local neighbor grids around target pedestrian.

2.3 Algorithms

2.3.1 Social-LSTM

Research works in human trajectory prediction focus on modeling human interactions to help predict future trajectory. Social-LSTM [3] applies social pooling mechanism (Figure 2.1) on LSTM to encode human interactions and predict next position at each timestep. Social pooling assumes interactions only happen in local neighbor grids around target pedestrian. For all pedestrians that fall into same grid, social pooling will simply add them element-wise to obtain fix sized vector for future prediction.

2.3.2 Social-Attention

As oppose to making the local interaction assumption as social-LSTM does, Social-Attention [45] considers all the pedestrians that share the scene and predict attention weights of target pedestrian to all other pedestrians. As shown in Figure 2.2, the interactions are modeled by a factor graph. Node factor encode information of current agent's position, spatial edge factor encode information of spatial relationship between two pedestrians, and temporal edge factor encode the past information of a pedestrian. When predicting the future trajectory for agent i, it's temporal edge information and the spatial edge information between agent i and another agent j is jointly considered to predict attention weight α_{ij}. The final prediction is then made by applying attention mechanism.

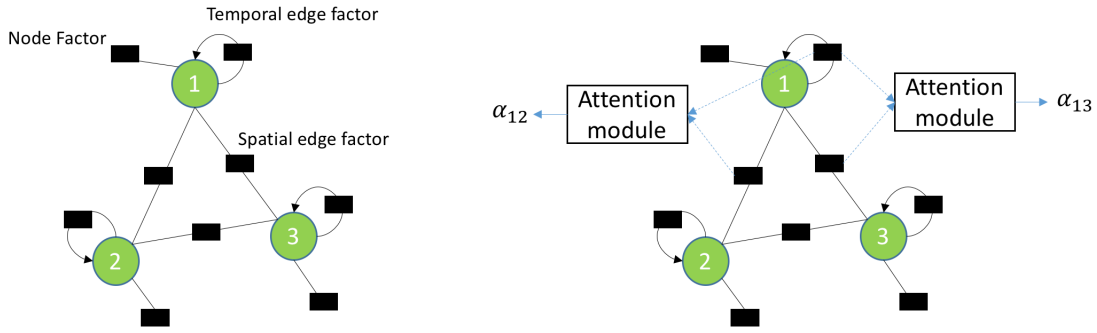

Figure 2.2: The interactions are modeled by a factor graph and attention weights are predicted using target pedestrian's temporal edge factor and the spatial edge factor between it and the other pedestrian.

2.3.3 Social-GAN

Both social-LSTM and social-Attention use regression loss against ground truth for training. However, the ground truth might not be the only trajectory that satisfy socially compliance constraint. There can be multiple alternatives. Therefore, Social-GAN introduces adversarial training into the loop. For a social generator that is capable of predicting human trajectory, they take the generated trajectory and forward it to a social discriminator to tell whether the trajectory is generated by generator or ground truth. With the adversarial training, the regression loss is no longer the only objective to be optimized and therefore it tries to learn for a social generator that is not merely optimized to match exact ground truth locations.

2.4 Conclusions

Research in human trajectories prediction inspires lots of works in socially compliant navigation in terms of how to model the interactions between humans and robot. Current works in human trajectories prediction focus on optimizing the similarity between predicted trajectory and human trajectories in low level behavior patterns. This is related to the *naturalness* aspect of socially compliant navigation. *Comfort* and *Sociability* aspects are not explicitly handled. The fundamental difference between human trajectories prediction and socially compliant navigation is that

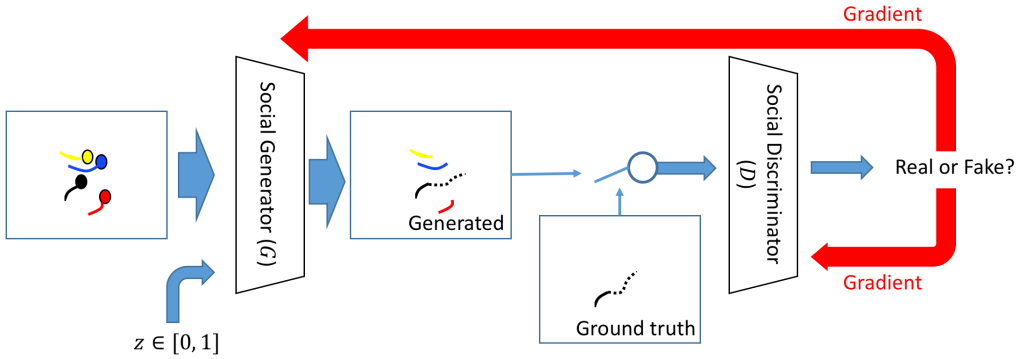

Figure 2.3: Social-GAN introduce adversarial training into the loop. The gradients from adversarial training help shape the social generator into inducing better prediction that is not merely optimized to match exact ground truth locations.

socially compliant navigation is generating navigation plan toward a specific goal point while human trajectories prediction is predicting trajectory for target agent without knowing the goal point.

Chapter 3

Literature Review - Socially Compliant Navigation

The two main stream approaches in the literature of socially compliant navigation are reinforcement learning approaches and inverse reinforcement learning approaches. The reinforcement learning approaches optimize the policy of the robot to maximize the expected reward defined by some metrics. The inverse reinforcement learning approaches learn the reward function from the recording of export demonstration to induce a policy that mimics the export behaviors. In the following sections, we will introduce state-of-the-art methods in both fields and conclude the challenges and limitations of such approaches.

3.1 Problem Definition

Let $\mathcal{D}$ denote a set of agents (pedestrians) who are involved in the time sequence $\{1, \ldots, T_{obs}, \ldots, T_{end}\}$. Given target agent i in $\mathcal{D}$, let g_i denote the goal state of agent i; $X_i = \{x_i^1, \ldots, x_i^{T_{obs}}\}$, the past state sequence of agent i for time steps $t = 1 \ldots T_{obs}$. For each agent i in $\mathcal{D}$, given a sequence of past trajectories of all agents $\{X_j : j \in \mathcal{D}\}$ and the target agent's goal state g_i, we aim to generate a navigation state sequence $\hat{Y}_i = \{\hat{y}_i^{T_{obs}+1}, \ldots, \hat{y}_i^{T_{end}}\}$ up to time step T_{end} with an objective of reaching the agent's goal state g_i in a socially compliant manner.

When we take *learning from demonstration* approaches, $Y_i = \{y_i^{T_{obs}+1}, \ldots, y_i^{T_{end}}\}$,

the demonstrated navigation state sequence provided by agent i is available, where x_i^t and $y_i^{t'}$ denote the state of agent i at time step t and t'.

3.2 Reinforcement Learning Approaches

The purpose of reinforcement learning is to find a "policy" that optimizes the expected "reward". The decision making process of a socially compliant navigation algorithm can be formulated as a Markov Decision Process (MDP) [39], which is a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{T}, r, \gamma)$ where $\mathcal{S}$ is the set of all possible states (e.g., all possible combinations of social agents and target robot and their position, velocity); $\mathcal{A}$ is the set of all possible actions (e.g., all possible rotation and translation a target robot can execute); T is the state transition probability function (e.g., the prediction of where each social agent will be in the next timestep); r is the reward function (e.g., Equation 1.1); and γ is a discount factor to control how greedy should the optimal policy be toward immidiate reward. A policy π is therefore a function of probability of action given current state at time t.

$$\pi(a|s) = \mathrm{P}[A_t = a|S_t = s]$$

3.2.1 LM-SARL

Chen et. al. [8] formulate their action space to be the heading and velocity of target robot. The optimal policy π^* they use is a deterministic policy:

$$\pi^*(s_t) = \underset{a_t}{\operatorname{argmax}}\, R(s_t, a_t) + \gamma^{\Delta t} \int_{s_{t+\Delta t}} \mathrm{T}(s_t, a_t, s_{t+\Delta t}) V^*(s_{t+\Delta t}) ds_{t+\Delta t} \tag{3.1}$$

where Δt is the decision interval, R is the reward function shown in Equation 1.1, $\mathrm{T}(s_t, a_t, s_{t+\Delta t})$ is the transition probability. and state s_t is the joint observation of both robot and social agents. V^* is the optimal value function:

$$V^*(s_{t+\Delta t}) = \sum_{t'=t}^{T} \gamma^{t'} R_{t'}(s_t, \pi^*(s_{t'}))$$

where T is the prediction horizon. However, V^* is intractable when our state and action space are huge. Therefore, they use a neural network to approximate V^* with

Algorithm 1 Deep V-learning

procedure DEEP V-LEARNING(D)
  Initialize value network V with demonstration D
  Initialize target value network $\hat{V} \leftarrow V$
  Initialize experience replay memory $E \leftarrow D$
  for episode = 1 to M **do**
    Initialize random state s_0
    repeat
      $a_t \leftarrow \text{argmax}_{a_t \in \mathcal{A}} R(s_t, a_t) + \gamma^{\Delta t} V(s_{t+\Delta t})$
      Store tuple $(s_t, a_t, r_t, s_{t+\Delta t})$
      Sample tuples $\{(s_i, a_i, r_i, s_{i+\Delta i}) : i = 1 \dots \text{batch size}\}$ from D
      Set target $y_i = r_i + \gamma^{\Delta i} \hat{V}(s_{i+\Delta i})$
      Update value network V by gradient descent
    until s_t reaches terminal state or $t \geq t_{max}$
    Update target network $\hat{V} \leftarrow V$
  end for
  return V
end procedure

Deep V-learning (Algorithm 1).

The function approximator for V^* should be designed to observe interactions between social agents and the robot and predict optimal value for the given interactions. To encode pair-wise feature between the robot and each human i, they feed a context vector s for the robot, context vector w_i for the human, and the result of social pooling (section 2.3.1) M_i around human i, through the MLP part of a LSTM to get e_i and hidden state h_i. The process is illustrated in Figure 3.1. To predict V value, attention weight α_i for human i is computed by a MLP that consider both e_i and their mean value e_m and the V value is then computed base on weighted average of all h (Figure 3.2).

During V-learning and testing, they use ORCA [43], a multi-robot collision avoidance algorithm, to simulate behavior of pedestrians (i.e., to serve as the T function in Equation 3.1).

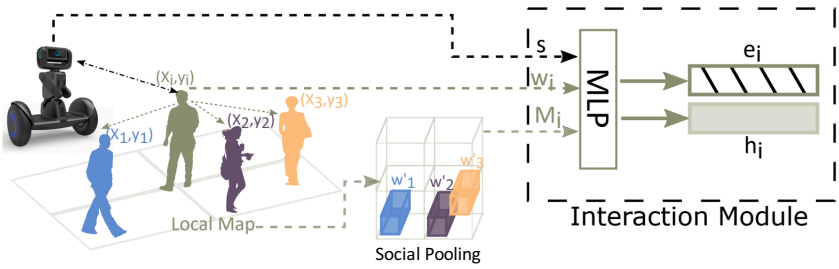

Figure 3.1: The joint feature (e_i, h_i) for human i and the robot is extracted by feeding a context vector s for the robot, context vector w_i for the human, and the result of social pooling M_i around human i, through a MLP.

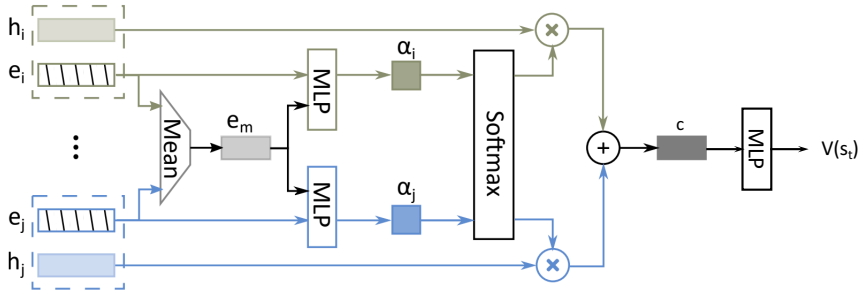

Figure 3.2: Attention weight α_i for human i is predicted by a MLP that consider both e_i and their mean value e_m. The V value is then computed base on weighted average of all h.

3.2.2 SA-CADRL

Similar to LM-SARL, Chen et. al. [9] use deep V-learning with hard coded social norms as reward function to generate socially-aware policy. The reward they designed consider three sets of interaction, passing, crossing, and overtaking, with bias toward right-handed rules (Figure 3.3). The reward is as following:

$$
\begin{aligned}
&R(s,a) = -I(s \in \mathcal{B}_{norm}) \\
&\quad \text{s.t. } \mathcal{B}_{norm} = \mathcal{B}_{pass} \cup \mathcal{B}_{ovtk} \cup \mathcal{B}_{cross} \\
&\qquad \mathcal{B}_{pass} = \{s | d_g > 3, 1 < \tilde{p_x} < 4, -2 < \tilde{p_y} < 0, |\tilde{\phi} - \psi| > 3\pi/4\} \\
&\qquad \mathcal{B}_{ovtk} = \{s | d_g > 3, 0 < \tilde{p_x} < 3, 0 < \tilde{p_y} < 1, |\tilde{\phi} - \psi| < \pi/4, |v| > |\tilde{v}|\} \\
&\qquad \mathcal{B}_{cross} = \{s | d_g > 3, \tilde{d_a} < 2, \tilde{\phi}_{rot} > 0, -3\pi/4 < \tilde{\phi} - \psi < -\pi/4\}
\end{aligned}
\tag{3.2}
$$

I is the indicator function. $\mathcal{B}_{pass}, \mathcal{B}_{ovtk}, \mathcal{B}_{cross}$ are the set of states satisfying hand crafted social rules of passing, overtaking, and crossing respectively. d_g is the agent's distance to goal, $\tilde{d_a}$ is the distance to the other agent, $\tilde{\phi}$ is the other agent's orientation, ψ is the agent's own orientation. $\tilde{\phi}_{rot}$ is the relative angular velocity between two agents, v is the velocity of target agent and $\tilde{v}$ is the velocity of the other agent. $\tilde{p_x}, \tilde{p_y}$ is the x,y coordinate of the other agent in target agent's frame. Therefore, $\mathcal{B}_{pass}$ can be understand as: when far away from goal ($d_g > 3$), two agents have near opposite heading ($|\tilde{\phi} - \psi| > 3\pi/4$), we penalize the state if the other agent is in the right-handed region of our target agent ($1 < \tilde{p_x} < 4, -2 < \tilde{p_y} < 0$). Similarly, $\mathcal{B}_{ovtk}$ can be understand as: when far away from goal, two agents have similar heading, and our target agent's speed is faster than the other ($|v| > |\tilde{v}|$), we penalize the state if the other agent is in our left. $\mathcal{B}_{cross}$ can be understand as: when far away from goal, two agents go close to each other ($\tilde{d_a} < 2$) with velocity toward different direction ($\tilde{\phi}_{rot} > 0$), we penalize states with not right-handed heading ($-3\pi/4 < \tilde{\phi} - \psi < -\pi/4$).

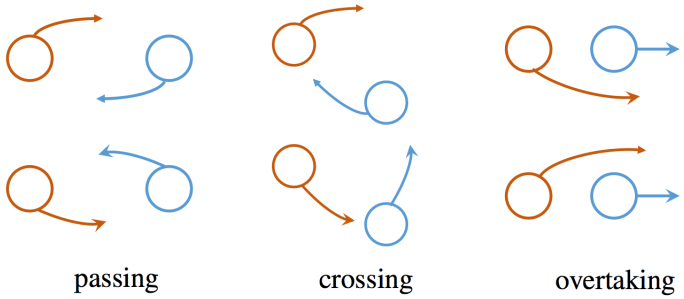

Figure 3.3: Left to right show two equally time-efficient ways for the red agent to pass, cross, and overtake the blue agent. The top row is left-handed rules and the bottom row is right-handed rules.

3.3 Learning from Demonstration Approaches

Instead of using reinforcement learning to find the optimal policy under some hand-crafted reward function, another main stream of approaches toward socially compliant navigation try to find the optimal policy directly from expert demonstration (which in our case, the trajectories generated by actual humans traveling among crowds). More specifically, [23, 25, 33, 35, 44] use inverse reinforcement learning technique to learn the reward function and optimal policy at the same time from demonstrations.

Inverse reinforcement learning assumes a MDP world whose reward function is not observable. Since finding a general form solution for reward function is very difficult, most IRL approaches assume it to be a linear combination of the state features. For a given state s, the reward can be expressed as the dot product $\Phi_s \cdot W$ of a feature vector Φ_s of the state and a weight vector W. In this case, the inverse reinforcement learning problem becomes finding the weight vector that induces the expert demonstrations as its optimal policy.

There are two main streams of inverse reinforcement learning algorithms, namely, max-margin IRL [1] and maximum entropy IRL [47]. They are based on a statistic F on set of state feature vectors which is used to compare the similarity between demonstrated states and the state sequences we obtained when applying optimal policy for a given weight estimate. Algorithm 2 outline the general procedure of both IRL methods where D is the set of states from expert demonstrations. Max-margin IRL uses the feature expectation as statistic F. And the weight update (line 10 of

Algorithm 2 Inverse Reinforcement Learning

```
procedure INVERSE REINFORCEMENT LEARNING(D)
    Set timestamp t ← 1
    Propose an initial weight vector Ŵ_1
    loop
        Compute the optimal policy π_t for the current weight vector Ŵ_t
        if Statistics similar enough (e.g., ‖F(D) − F(π_t)‖ < ε) then
            return Ŵ_t
        else
            t ← t + 1
            Update Ŵ_t using algorithm-dependent method
        end if
    end loop
end procedure
```

the algorithm) is done by maximizing the difference between all the previously found expected rewards and the demonstrated expected rewards. Maximum entropy IRL use feature sum as the statistic. Their weight update is a probabilistic approach based on the principle of maximum entropy.

As inverse reinforcement learning algorithms estimate reward function directly from expert demonstrations, researches on inverse reinforcement learning for socially compliant navigation mostly focus on finding good features that capture the social interaction to represent states. The principle of those features can be categorize into 4 types of features.

1. **Density features:** the main purpose of these features is to encode the local density in the neighborhood of the target agent (e.g., the number of agents within a certain radius around the target agent).
2. **Speed and orientation features:** these features represent the relative speed and orientation of each social agent with respect to the robot. For example, in Figure 3.4, we compute speed and orientation features in three orientations – O_1 toward, O_2 perpendicular to, or O_3 away from the robot; and three speeds – S_1 slow, S_2 medium, or S_3 fast.
3. **Social force features:** these features are inspired by the social force model (Equation 2.1) and they capture the interactions among other agents that share the scene with the robot.

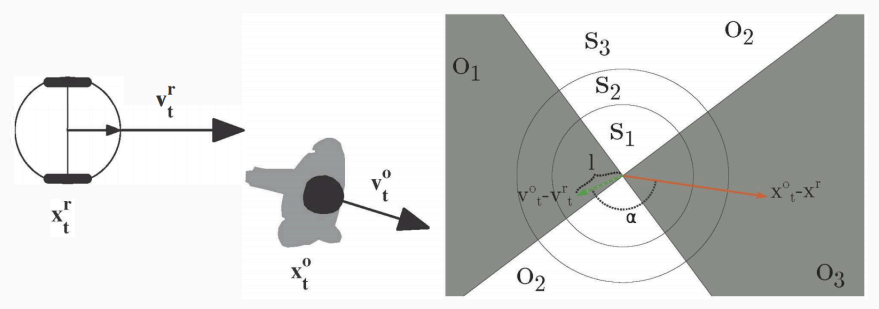

Figure 3.4: Speed and orientation features represent the relative speed and orientation of each social agent with respect to the robot.

4. **Default cost feature:** this feature is always set to one. It has been proposed to allow IRL to learn how people balance the other features agianst the travelled distance.

3.4 Conclusions

Reinforcement learning approaches use a reward function to guide the learning of an optimal policy. The reward function of LM-SARL (Equation 1.1) covers only the *comfort* aspect of socially compliant navigation. It is merely trying to perform dynamic obstacles collision avoidance with pedestrians as obstacles. The hand-crafted penalty function of SA-CADRL (Equation 3.2) tries to cover both comfort and naturalness but only on three hand-picked types of interaction. As the navigation decisions made by a pedestrian is complex and the objective is implicit, a comprehensive reward function to cover both comfort and naturalness for general cases is extremely hard to define.

Acquiring episodes for training a reinforcement learning model can be expensive. To the best of our knowledge, none of the works in socially compliant navigation have used actual pedestrians for rolling out episodes during training. Instead, Most of the works use simulated pedestrians during training. As there is not a perfect pedestrian simulator available, training with simulated pedestrians can lead to huge paradigm shift between training and real-world deployment.

Inverse reinforcement learning approaches try to cover *naturalness* by learning

the reward function and policy directly from human demonstrations to mimic the low level behavior patterns of a human. However, the framework of inverse reinforcement learning only encourages the induced policy to mimic the expert demonstrations which are all positive examples. This makes the learned policy hard to generalize during real-world deployment due to the lack of avoiding undesired behavior during training. For example, the expert demonstrations of human trajectories tend not to collide with each other. During training, the model learns from those collision-free demonstrations and does not get penalty from producing colliding behavior.

Also, research works in inverse reinforcement learning for socially compliant navigation requires careful feature engineering. The performance of the resulting model is therefore limited by the quality of the features. As the human traversal decision is complex and implicit, perfect features are hard to acquire.

We therefore conclude the challenges of reinforcement learning and inverse reinforcement learning approaches:

- Reinforcement learning
  - Hard to specify sophisticated reward function that leads to a good policy for both comfort and naturalness.
  - Hard to roll out episodes for training.
    - Real world execution is expensive and not safe for social agents.
    - Perfect pedestrian simulator does not exist. Can lead to huge paradigm shift between training and testing.
- Inverse reinforcement learning
  - No demonstration of negative samples makes the learned policy hard to generalize.
  - Require careful feature engineering.

Chapter 4

A Generative Approach for Socially Compliant Navigation

With interpretability in mind, we focus on developing a model that covers both *comfort* and *naturalness* aspects of socially compliant navigation.

Following the theory of social force, we propose NaviGAN, a generative adversarial network architecture for social navigation, that represents an agent's intention to reach a goal while trying to comply with social norm of crowd navigation behavior. To prioritize the social aspects of the navigation, we omit the estimation of $\vec{F}_{obs}$ in Equation (2.1) as in the baseline approaches.

4.1 NaviGAN

An overview of our proposed model is shown in Figure 4.1. NaviGAN consists of three building blocks: (i) Intention-force generator that models an agent's intention for reaching a destination, (ii) Social-force generator that models the social force $\vec{F}_{social}$ and fluctuation force $\vec{F}_{fluct}$, and (iii) Social-aware discriminator that discover the latent social influences from discriminating those navigation paths generated by the generator against expert demonstrations. Each building block is described in detail in Section 4.1.1–Section 4.1.3.

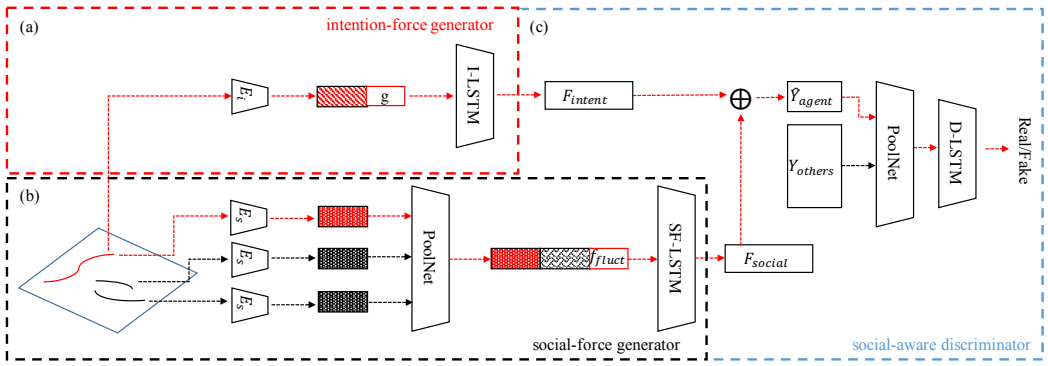

Figure 4.1: Overview of the proposed model. NaviGAN is composed of three major building blocks: 1. Intention-force generator (block (a), section 4.1.1) that models the intention force $\vec{F}_{intent}$ toward goal state; 2. Social-force generator (block (b), section 4.1.2) that models the social force $\vec{F}_{social}$ and fluctuation force $\vec{F}_{fluct}$; and 3. Social-aware discriminator (block (c), section 4.1.3) that discovers the latent social aspects from discriminating between expert demonstrations and navigation behavior generated by the generator. The red color indicates that the vector holds data relevant to the target agent, and the black color is used for other agents.

4.1.1 Intention-Force Generator

The purpose of the intention-force generator is to generate a path (or a sequence of states) $\tilde{Y}_i = \{\tilde{y_i}^{T_{obs}+1}, \ldots, \tilde{y_i}^{T_{end}}\}$ toward goal state g_i, conditioned on goal g_i and the past state sequence $X_i = \{x_i^1, \ldots, x_i^{T_{obs}}\}$ of the target agent. Note that this part represents self-interested decision making where the benefits of other agents are not considered. As shown in block (a) of Figure 4.1, the intention-force generator follows an encoder-decoder design. We encode the past state sequence X_i of target agent i using an encoder LSTM E_i. The initial hidden state h_i^0 and cell state c_i^0 of the target agent are initialized with a zero vector. For time steps t from 1 to T_{obs}, we perform the following update operation to encode the input sequence:

$$[h_i^t; c_i^t] = E_i(x_i^t, [h_i^{t-1}; c_i^{t-1}]), \tag{4.1}$$

where the operation $[u; v]$ denotes the operation of concatenating vectors u and v. We then inject goal state g_i to the resulting hidden state to initialize the hidden state

$\mathrm{h}_i^{T_{obs}}$ for the decoder, known here as Intention-LSTM (I-LSTM), as follows:

$$\mathrm{h}_i^{T_{obs}} = [h_i^{T_{obs}}; g_i].$$

For timesteps t from $T_{obs}+1$ to T_{end}, the decoding process can then be described by the following equations:

$$[\mathrm{h}_i^t; \mathrm{c}_i^t] = \text{I-LSTM}(\tilde{y}_i^{t-1}, [\mathrm{h}_i^{t-1}; \mathrm{c}_i^{t-1}])$$
$$\tilde{y}_i^t = \mathrm{NN}_{spatial}(\mathrm{h}_i^t),$$

where $\tilde{y}_i^{T_{obs}}$ is initialized with $x_i^{T_{obs}}$. $\mathrm{NN}_{spatial}$ is a simple feed forward neural network to map the hidden state of I-LSTM h_i^t to state prediction output $\tilde{y}_i^t$.

4.1.2 Social-Force Generator

The purpose of the social-force generator is to model the social force F_{social} and fluctuation force F_{fluct}. A diagram of the social-force generator can be found in block (b) of Figure 4.1.

Social Force: Given all of the observed state sequences $X_{\mathcal{D}} = \{X_j : j \in \mathcal{D}\}$, the social-force generator will generate a sequence of social forces $\{\vec{f}_i^{T_{obs}+1}, \ldots, \vec{f}_i^{T_{end}}\}$. The composite social force $\vec{f}_i^t$ is the ensemble of social force and fluctuation force that models the social responses of the target agent to all of other agents in the scene as well as the stochastic nature of human navigation policies. The social-force generator is also an encoder-decoder design. Whereas the Intention-Force generator encodes only the target agent state sequence and injects a goal state right after encoding, the Social-Force generator performs encoding for all of the agents in the scene and then utilizes a special pooling mechanism before decoding. The idea of designing the social-force generator is to predict the social interactions between agents purely based on their relative positions and previous state sequences. We encode all of the observed state sequences $X_j \in X_{\mathcal{D}}$ using an encoder LSTM E_s. For simplicity, we reuse the notations of h and c to represent the hidden state and cell state related to E_s; h and c to represent the hidden state and cell state of SocialForce-LSTM (SF-LSTM). The initial hidden state h_j^0 and cell state c_j^0 of agent j are initialized with a zero vector. Using the same encoding process as Equation (4.1), we obtain the

encoding vector $h_j^{T_{obs}}$ of each X_j.

For the efficiency and simplicity, we adopt a similar pooling mechanism as [15]. The *PoolNet* module will consider the displacement–the vector from target agent's coordinate to other agent's coordinate–from the target agent to all of other agents when generating a pooled context vector that is composed of encoding vectors of all other agents and the target agent. Let $d_{ij}^{T_{obs}}$ denote the displacement from target agent i to agent $j \in \mathcal{D} \setminus \{i\}$ at time T_{obs}. We obtain a displacement-sensitive context embedding $e_{ij}^{T_{obs}}$ for each X_j by: 1) feeding displacement $d_{ij}^{T_{obs}}$ through a position encoding neural network NN_{pos}, and then 2) concatenating the output with $h_j^{T_{obs}}$ and feeding through the displacement-sensitive embedding network NN_{embed}. The process is formally described in the following equation:

$$e_{ij}^{T_{obs}} = \mathrm{NN}_{embed}([h_j^{T_{obs}}; \mathrm{NN}_{pos}(d_{ij}^{T_{obs}})]).$$

After we obtain the displacement-sensitive context embedding $e_{ij}^{T_{obs}}$ for each agent, we aggregate them into a single vector $\mathrm{V}_i^{T_{obs}}$ using max pooling as defined as follows:

$$\mathrm{V}_i^{T_{obs}}[idx] = \max_{j \in \mathcal{D} \setminus \{i\}} e_{ij}^{T_{obs}}[idx], \tag{4.2}$$

where $v[idx]$ denotes the indexing operation that indexes the idx^{th} entry of vector v. The final vector $\mathrm{V}_i^{T_{obs}}$ is the ensemble of all of the information from other agents that leads the social-force generator to determine the target agent's social responses to other agents.

Fluctuation Force: Finally, we initialize the hidden state $\mathrm{h}_i^{T_{obs}}$ of the SF-LSTM by concatenating the target agent's context vector $h_i^{T_{obs}}$, the aggregated context vector $\mathrm{V}_i^{T_{obs}}$, and a random noise vector f_{fluct} sampled from $\mathcal{N}(0, 1)$ which will serve as the random seed for modeling the stochastic $\vec{F}_{fluct}$.

$$\mathrm{h}_i^{T_{obs}} = [h_i^{T_{obs}}; \mathrm{V}_i^{T_{obs}}; f_{fluct}]$$

The cell state $\mathrm{c}_i^{T_{obs}}$ is, again, initialized with a zero vector. For time steps t from

$T_{obs}+1$ to T_{end}, the decoding process can then be described by the following equations:

$$[\mathrm{h}_i^t; \mathrm{c}_i^t] = \text{SF-LSTM}(\vec{f}_i^{t-1}, [\mathrm{h}_i^{t-1}; \mathrm{c}_i^{t-1}])$$
$$\vec{f}_i^t = \mathrm{NN}_{social}(\mathrm{h}_i^t),$$

where $\vec{f}_i^{T_{obs}}$ is initialized with a zero vector and NN_{social} is a feed forward neural network to map the hidden state of SF-LSTM h_i^t to social force prediction $\vec{f}_i^t$.

4.1.3 Social-Aware Discriminator

The way-point $\hat{y}_i^t$ that the target agent should reach by executing an action at time t is decided by combining the intention force and social forces:

$$\hat{y}_i^t = \tilde{y}_i^t + \vec{f}_i^t.$$

Based on the intuition that there can be multiple optimal behaviors that may not be present in expert demonstrations, we take the adversarial training approach as opposed to mimicking demonstrations by optimizing on the negative likelihood of demonstrations. Social-GAN [15] shares a similar intuition to ours; however, their discriminator takes only the state sequence of target agent $X_{real} = [X_i; Y_i]$ or $X_{fake} = [X_i; \hat{Y}_i]$ as input, whereas our discriminator takes all of the agents in the scene as input and highlights the target agent. More specifically, the input to our discriminator is $X_{real} = \{[X_i; Y_i]\} \cup \{[X_j; Y_j] : j \in \mathcal{D} \setminus \{i\}\}$ or $X_{fake} = \{[X_i; \hat{Y}_i]\} \cup \{[X_j; Y_j] : j \in \mathcal{D} \setminus \{i\}\}$. We argue that augmenting the input to cover other agents is of crucial importance since most of the latent social aspects are established on the interaction between target agent and others. The prediction process of social-aware discriminator is illustrated in block (c) of Figure 4.1. Assume the input target agent state sequence is $\{s_i^t : t = 1 \dots T_{end}\}$, and other agent state sequence is $\{s_j^t : t = 1 \dots T_{end}\}$. At each time step t we aggregate the target agent state s_i^t with other agents' state s_j^t into one target-centric, displacement-sensitive context embedding v_i^t using the PoolNet module as described in Section 4.1.2. The sequence of target-centric embedding $\{v_i^t : t = 1 \dots T_{end}\}$ is then fed to Discriminator-LSTM (D-LSTM) to encode the sequence into one unified vector $h_i^{T_{end}}$ which is the last hidden state of D-LSTM. Base on $h_i^{T_{end}}$, the discriminative score is predicted

to identify whether the input sequence is generated by expert demonstration or generators.

Latent Social Aspects Discovery

Without adversarial training, the $L2$ objective would lead the generators toward cloning the behavior by only matching the demonstrated states. This tendency to overfit the expert demonstrations can result in learning a policy that merely reproduces the demonstrated sequences without understanding the latent social aspects at a higher level. In our setting, generator G is the function that produces the joint output of I-LSTM and SF-LSTM, and discriminator D is a social-aware discriminator. We, therefore, modify the GAN objective in Equation (2.2) to the following:

$$\begin{aligned}\min_G \max_D \mathop{\mathbb{E}}_{S_i, S_\mathcal{D}} [\log D(S_i, S_\mathcal{D})] + \\ \mathop{\mathbb{E}}_{S_i, S_\mathcal{D}, f_{fluct}} [\log(1 - D([X_i; G(X_i, S_\mathcal{D}, f_{fluct})], S_\mathcal{D}))].\end{aligned} \tag{4.3}$$

To estimate the expectation, we first sample the state sequences of all agents $S_\mathcal{D} = \{[X_j; Y_j] : j \in \mathcal{D}\}$ in a scene, from the entire demonstration set $\mathcal{S}$. Then we sample the target agent state sequence $S_i = [X_i, Y_i]$ from $S_\mathcal{D}$.

4.1.4 Enforce Comfort

The adversarial training framework discovers latent social aspects to guide generators into producing policy that mimics the low level behavior patterns demonstrated by the recording. Though this covers the *naturalness* aspect of socially compliant navigation, the *comfort* aspect is not explicitly governed. Therefore, we include resistance loss $\mathcal{L}_{resist}$ that explicitly penalizes behaviors where predicted trajectory comes closer to other pedestrian than a safety distance threshold.

$$\mathcal{L}_{resist} = \|\max(d_{safe} - d_o, 0)\|_2 \tag{4.4}$$

d_{safe} is the safety distance threshold, and d_o is the distance from target agent to other agent. By including $\mathcal{L}_{resist}$ in our training framework, we jointly optimize both *comfort* and *naturalness*.

Metric	ADE					
Dataset	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
Linear	1.33	0.39	0.82	0.62	0.77	0.79
Vanilla-LSTM	1.09	0.86	0.61	0.41	0.52	0.70
Social-LSTM	1.09	0.79	0.67	0.47	0.56	0.72
Social-Attention	**0.39**	**0.29**	**0.33**	**0.20**	**0.30**	**0.30**
Social-GAN	1.13	1.01	0.60	0.42	0.52	0.74
Goal-S-LSTM	1.02	0.44	0.85	0.42	0.46	0.64
NaviL2	0.97	0.47	0.70	0.42	0.43	0.60
NaviL2†	0.99	0.49	0.93	0.41	0.51	0.67
NaviGAN	0.95	0.43	0.85	0.40	0.47	0.62
NaviGAN†	0.97	0.44	0.93	0.42	0.50	0.65
NaviGAN-R	1.53	1.22	1.67	0.72	0.78	1.18
NaviGAN-R†	1.62	1.17	1.35	0.75	0.77	1.13

Table 4.1: Quantitative results of all methods. Average Displacement Error is reported in meters with $T_{obs} = 8$ and $T_{pred} = 12$. Our methods consistently outperform others except for Social-Attention because of ours longer prediction horizon as discussed in section 4.1.5.

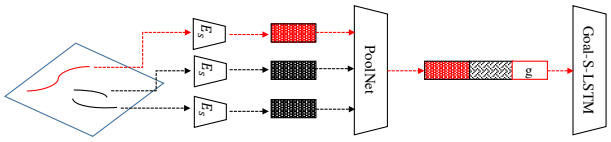

Figure 4.2: The architecture of Goal-Social-LSTM. Goal state information g is directly injected after PoolNet layer to predict state sequence condition on goal state.

4.1.5 Experiments

For the consistency throughout the experiments, we set the dimension of hidden states of all of the encoder and decoder LSTMs to 32. We iteratively train the generators and discriminator with batch size 32, the number of target agent for each batch, for 500 epochs using the Adam [22] optimizer with learning rate of 0.001.

We conduct our experiments on two publicly available datasets: ETH [34] and UCY [27]. In total, there are 5 sets of data (ETH, HOTEL, UNIV, ZARA1, ZARA2) with 4 different scenes and 1,536 pedestrians in crowded settings. Each pedestrian is associated with a unique index, annotated at 2.5 fps, to track its position in real world coordinates (meters in x-y direction) through time. We set the $T_{obs} = 8$

Metric	FDE					
Dataset	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
Linear	2.94	0.72	1.59	1.21	1.48	1.59
Vanilla-LSTM	2.41	1.91	1.31	0.88	1.11	1.52
Social-LSTM	2.35	1.76	1.40	1.00	1.17	1.54
Social-Attention	3.74	2.64	3.92	**0.52**	2.13	2.59
Social-GAN	2.21	2.18	1.28	0.91	1.11	1.54
Goal-S-LSTM	1.73	0.8	1.35	0.69	0.70	1.05
NaviL2	1.73	0.88	**1.19**	0.68	**0.66**	1.03
NaviL2†	1.73	0.75	1.51	0.65	0.68	1.06
NaviGAN	**1.64**	0.74	1.36	0.66	0.72	**1.02**
NaviGAN†	1.65	**0.70**	1.53	0.69	0.70	1.05
NaviGAN-R	1.76	1.20	1.71	0.75	0.76	1.24
NaviGAN-R†	1.76	1.13	1.55	0.72	0.70	1.17

Table 4.2: Quantitative results of all methods. Final Displacement Error is reported in meters with $T_{obs} = 8$ and $T_{pred} = 12$. The FDE score is significantly improved.

(3.2 seconds) and $T_{pred} = T_{end} - T_{obs} = 12$ (4.8 seconds). To encourage a model to learn long-term, complex social behavior, we follow the common practice [3, 15] of selectively choosing the pedestrians that stay in the receptive field for longer than T_{pred} to be our target agents, and learn the policy through their demonstration. We note that the approaches using this idea can result in higher final displacement error (FDE) because FDE tends to grow larger in long-term predictions.

Coordinate frames: As we model the interactions between a target agent and other agents, we use a target-centric coordinate frame. We represent X_i, $X_{\mathcal{D}}$, Y_i, $\tilde{Y}_i$, and goal state g_i using the target-centric coordinate frame where its origin is on the point x_i^0 in world coordinates. In comparison to world coordinates, the use of target-centric coordinates tends to result in a more general model by preventing the model from learning location-specific biases.

Baselines: We compare the performance against following baselines:

1. *Linear*: A simple linear regressor that estimates linear parameters by minimizing the least squared error.
2. *Vanilla-LSTM*: LSTM without pooling mechanism.
3. *Social-LSTM* [3]: Local social behavior (within the neighbor grid) among agents is modeled using social-pooling on an LSTM-encoded vector of each agent.

4. *Social-Attention* [45]: The attention weights are parametrized through an S-RNN to make joint predictions that selectively consider all agents in the scene.
5. *Social-GAN* [15]: GAN-based training of an LSTM with PoolNet whose discriminator considers only the target agent but not others.

Proposed Models:

1. *Goal-S-LSTM*: Goal-Social-LSTM. A simplified version of NaviGAN trained with the L2 objective, denoted by $\mathcal{L}_2 = \|Y_i - \hat{Y}_i\|_2$. Goal-S-LSTM omits the intention-force generator and directly injects the goal state information to the decoder's initial hidden state right after the PoolNet layer to predict the state sequence. A diagram of Goal-S-LSTM is shown in Figure 4.2.
2. *NaviL2, NaviL2*†: The generators of NaviGAN (block (a) and (b) in Figure 4.1) trained with both $\mathcal{L}_2$ and *FDE loss* $\mathcal{L}_{fde}$. Notice that $\mathcal{L}_{fde}$ is applied on the prediction of intention-force generator ($\tilde{Y}_i$). As the final prediction $\hat{Y}_i$ is made by summing $\tilde{Y}_i$ and $\vec{F}_{social}$, it is unclear for the model to decide which module should learn the intention and which module should learn the social-awareness. The purpose of $\mathcal{L}_{fde}$ is to eliminate the ambiguity of such composite predictions. The use of the FDE loss guides the intention-force generator toward generating intention force. NaviL2 intuitively models $\vec{F}_{intent}$ and $\vec{F}_{social}$. We do not include fluctuation noise $\vec{F}_{fluct}$ when running the social-force generator for NaviL2.

$$\mathcal{L}_{fde} = \|Y_{T_{end}} - \tilde{Y}_{T_{end}}\|_2 \tag{4.5}$$

   NaviL2† denotes a model that has only the intention-force generator from a trained NaviL2. The plans generated by NaviL2† will thus be purely intention-force without considering other agents. The purpose of this intention-force only model is to visualize the target agent's intention for analysis.
3. *NaviGAN, NaviGAN*†*, NaviGAN-R, NaviGAN-R*†: The complete algorithm with social-aware discriminator trained with $\mathcal{L}_2$, $\mathcal{L}_{fde}$, and adversarial objective (Eq 4.3). Analogous to NaviL2†, NaviGAN† denotes the intention-force generator of NaviGAN. NaviGAN-R is the complete framework jointly trained with $\mathcal{L}_{resist}$. We set 0.5m as d_{safe} for resistance loss.

Short-term Human Trajectories Prediction

We demonstrate the ability of NaviGAN to plan a human-like path by showing its performance in human trajectories prediction. As in [3] and [45], we train and validate models on 4 sets, test on the remaining set, and report the average performance in Table 4.1 and Table 4.2.

Metrics: The **average displacement error (ADE)** computes the mean Euclidean distance between the predicted trajectory and true trajectory at each estimated point, whereas the **final displacement error (FDE)** measures the Euclidean distance between the predicted final location and true location. We select the state right before target agent leave the scene as its goal state and plan its future path for T_{pred} time steps.

The result shows significant improvement over other baselines especially in terms of FDE. By comparing the performance between Goal-S-LSTM and NaviL2, we can see how social force modeling helps the training process to produce consistently better results. When compared to NaviL2, the extra stochasticity of NaviGAN, which include $\vec{F}_{flunct}$ and the GAN objective, does not degrade the performance both in terms of FDE and ADE. With the resistance loss, NaviGAN-R sacrifice ADE and FDE to ensure *comfort*. However, this aspect is not observable by looking only at ADE and FDE. In section 4.1.5, we will provide extra metrics to measure the *comfort* aspect.

We note that Social-Attention does not selectively choose agents in evaluation, making predictions even for short-term pedestrians who exit the scene before T_{end}. These shorter predictions can help achieving a smaller ADE by excluding potentially severe compounding errors at a later time step.

Long-term Playback Navigation

We aim to demonstrate the ability of NaviGAN to perform long-term navigation using the playback of recorded data; therefore, we set the goal state to be the target agent's position $3 \cdot T_{pred}$ (14.4 seconds) into the future and roll out the episode with a cut-off horizon at $5 \cdot T_{pred}$ (24 seconds). During execution, whenever a target agent arrives within 0.5m from its goal state, we end the episode and mark the navigation as a success. At each time step, we use a target agent's previous predictions and

model	Goal	L2	L2†	NG	NG†	NG-R	NG-R†	human
S-score	0.40	0.38	0.40	0.41	0.40	0.38	0.38	**0.44**
comfort%	0.81	0.81	0.82	0.82	0.80	**0.97**	0.85	0.96
arrival%	0.91	0.88	0.92	0.97	0.92	0.85	0.88	**1.00**

Table 4.3: We separately look at the two factors of social score – arrival rate and comfort rate – to understand the performance of each model. For a compact layout, we denote Goal-S-LSTM as Goal, NaviL2 as L2, and NaviGAN as NG.

others' observed trajectories as input to predict the next action and execute in order to simulate a realistic navigation.

By the complex nature of human navigation, the L2 distance to recorded trajectories does not necessary reflect the social-awareness of a planning algorithm. Therefore, we seek other statistics to demonstrate the social-awareness. Our first attempt is to measure the social score as [8] does. With human in the recorded dataset scoring 0.44, All other algorithms score around 0.4 and there is no obvious difference between algorithms. We instead look at the two factors of social score – arrival rate and comfort rate – separately. Following their design, we set the comfort distance to 0.2 meter and arrival tolerance to 0.5 meter. Comfort rate is computed as the percentage of episodes who do not contain cases of robot getting closer to other pedestrians than the comfort distance. Arrival rate is computed as the percentage of episodes who contain cases of robot arriving at goal point within arrival tolerance before the cut-off horizon (24 seconds into the future).

The results are shown in Table 4.3. Without $\mathcal{L}_{resist}$, the models focus only on the *naturalness* aspect of socially compliant navigation and therefore, even with high arrival rate, they achieve lower comfort rate. By adding $\mathcal{L}_{resist}$, we optimize for both *naturalness* and *comfort*, and achieve 0.97 comfort rate which is even higher than the recorded human trajectories. By comparing the comfort rate between NaviGAN-R and NaviGAN-R*, we can clearly see the capability of social generator modifying prediction of intention generator to avoid uncomfortable behaviors. However, to enforce *comfort*, NaviGAN-R sometimes will sacrifice FDE and therefore result in lower arrival rate.

Interpretability: For better understanding of the relationship between the intention-force generator and the social-force generator, we visualize two examples

of episodes rolled out by NaviGAN and NaviGAN† in Figure 4.3 and Figure 4.7, respectively. In these examples, we can clearly see how the social-force generator adjusts the sequence predicted by the intention-force generator to enforce social-compliance, maintain comfortable distance to others, while still moving toward a goal state.

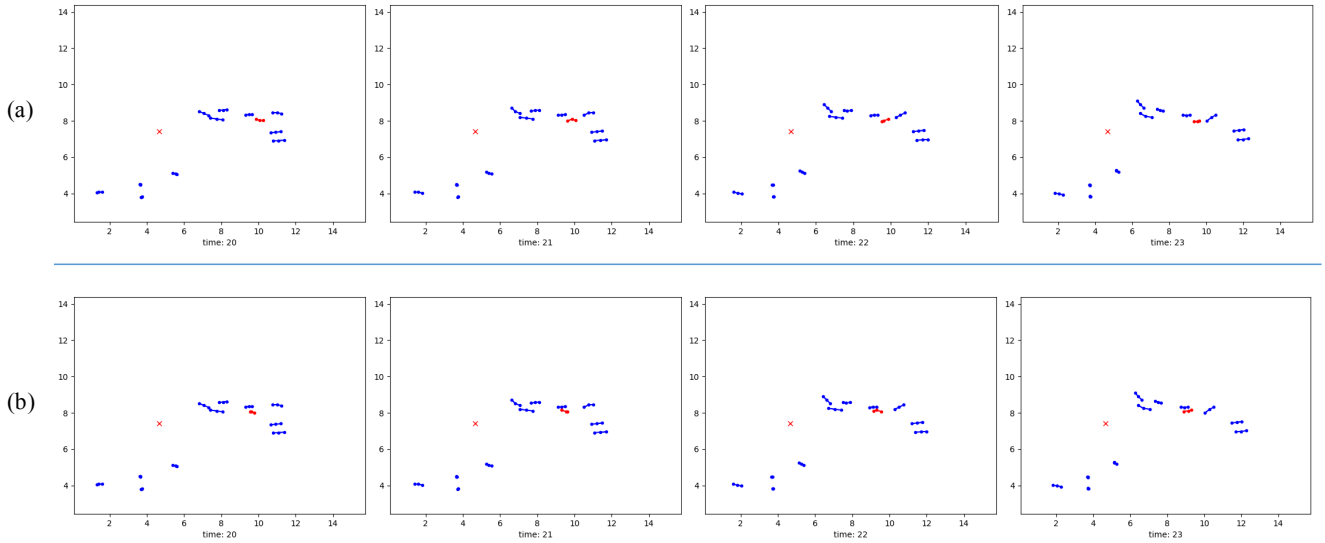

Figure 4.3: An example where the social-force generator affects the sequence predicted by the intention-force generator to enforce social norm such as avoiding to get uncomfortably close to others, while still pursuing a goal (marked as a red cross). We visualize the state sequence at time t by showing its two previous states along with the current state in a dotted-line. Red dots are the footprints of our target agent, and blue dots are the footprints of other agents. Subfigure (a) visualizes the path planned by NaviGAN, and (b) visualizes the path planned by NaviGAN† (intention-force only).

Simulated Navigation

In order to investigate the performance of NaviGAN-R qualitatively, we simulate an optimal robot who is capable of perfectly execute the next way point predicted by our model. We playback the testing set to obtain realistic pedestrian trajectories and select starting point and goal point to qualitatively visualize the predicted trajectory. The results are shown in figure 4.5, 4.6, and 4.7. Again, we visualize the state sequence

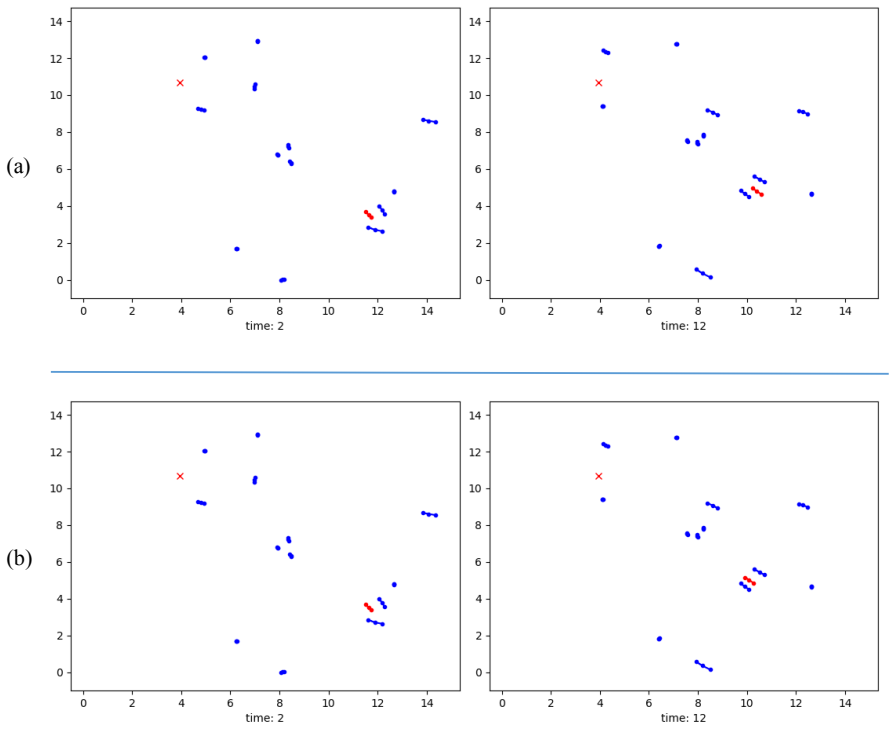

Figure 4.4: An example where $\vec{F}_{social}$ slows down the pace of the target agent to avoid getting to an uncomfortable distance to other agents, whereas $\vec{F}_{intent}$ is to reach the goal state faster. Subfigure (a) visualizes the path planned by NaviGAN, and (b) visualizes the path planned by NaviGAN†.

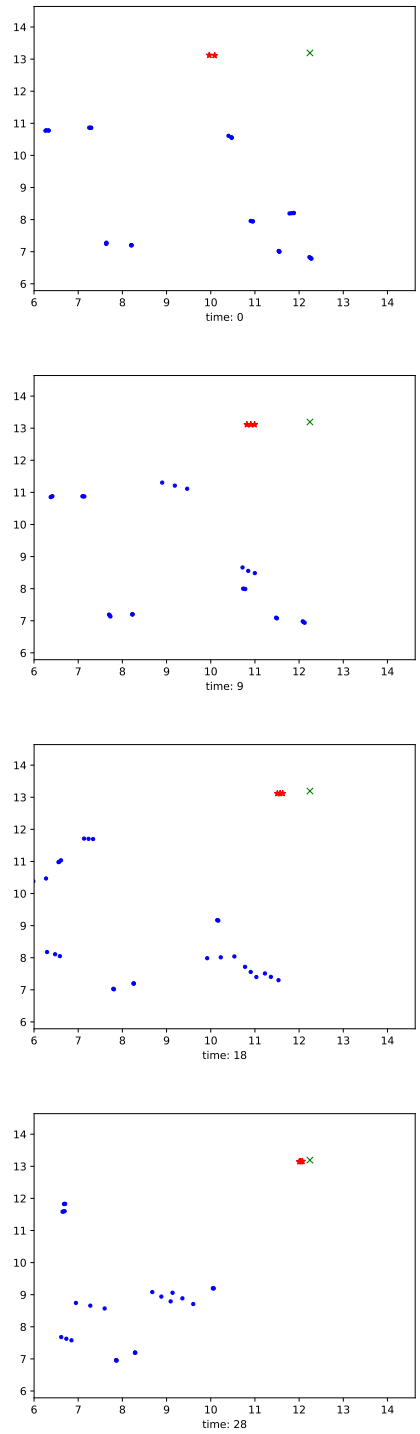

Figure 4.5: When clear from traffic, the predicted trajectory navigates directly to the goal point.

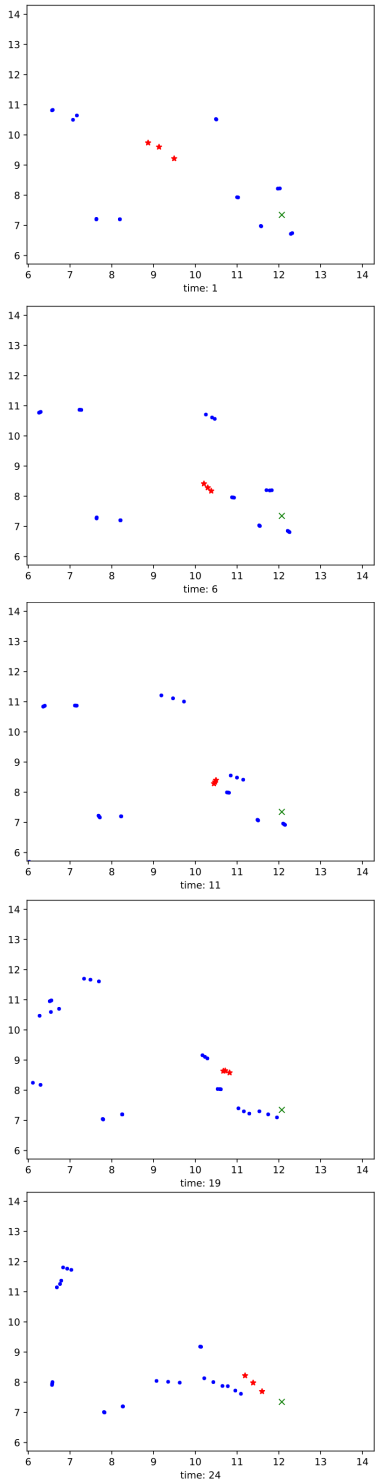

Figure 4.6: The predicted trajectory successfully weaves through dense crowd to reach the goal point.

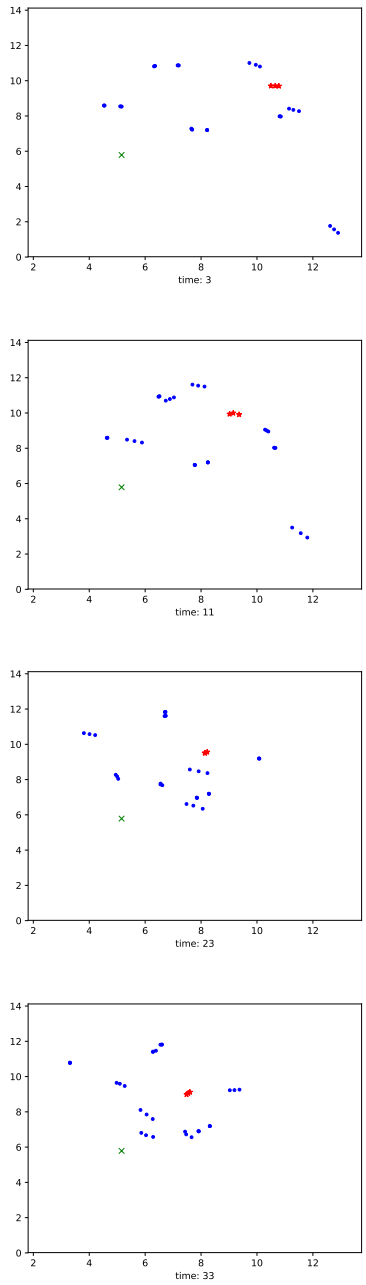

Figure 4.7: When facing dense crowd crossing in front of our target agent, the predicted trajectory slows down to let others pass first and then picks up speed after traffic clears off.

Figure 4.8: We conduct robotic experiments on a ClearPath Husky robot.

at time t by showing its two previous states along with the current state in dots. Blue dots are playbacks from the dataset, and red dots are predicted trajectory. Red cross marks the goal point of target agent.

4.2 Robot Experiments

We use a ClearPath Husky robot (figure 4.8) to provide mobility and deploy our model (NaviGAN-R) on a Nvidia Jetson TX2 development board. A lidar-based Kalmen filter [46] tracker is applied to identify and keep track of pedestrians in the scene. Beside qualitatively looking at each run of social navigation, we perform two set of experiments – qualitative measure and A/B testing – to investigate the performance of our approach.

4.2.1 Qualitative Measurement

As a qualitative measurement, we systematically design scenarios to interact with the robot running our socially compliant navigation module. We score each run with one of the three types of score:

- **score 1**: Collision with other pedestrian happens during run. The behavior is obviously not socially compliant.
- **score 2**: No collision happens. Robot demonstrates not natural behavior and only mild social compliance is observed.

- **score 3**: The robot complies with social rules in a natural and comfort manner.

The experimental design was carried out on JMP 13 software by SAS Inc. The experimental plan was structured as a Main effects design, completely randomized D-optimal design in 24 runs.

The inputs for the design were: Number of Pedestrians (2, 4, 6), Walking Direction (Parallel to the path of the oncoming robot, Parallel and Crossing the path, Crossing), and Crossing Distance (meters) that pedestrians crossed in front of the oncoming bit (1, 2, 3) in meters. Pedestrians was entered as "discrete numeric", Direction as "categorical", and crossing distance as "numeric". In "Parallel & Crossing" some pedestrians crossed the path of the robot (at approximately 1, 2, or 3 meters distance) while others (at outer edge) continued on a parallel path. Pedestrians tried to walk steadily forward and at arms length from each other.

We provide video (link: https://youtu.be/61blDymjCpw) of all trials for this experiment. The goal point is always set to be on the right of the orange cone. In all runs, the robot never collides with pedestrians (i.e., score 1 never happened) and it always reaches the goal point within 0.5 meter tolerance. Among all the trials, the algorithm get 86.7% of score 3, and 0% of score 1. In most of the failure cases (cases who score 2, e.g., experiment #5 and #9), unnatural behavior happens only after interaction (i.e., all pedestrians have passed) and probably is due to limitation of Husky's dynamic so the robot is spinning in-place before carrying on toward the goal point. Experiment #17 is a particularly interesting case where the robot actually roll back to avoid collision before the space clear off and it continues to navigate. This behavior is marked as unnatural and therefore get scored 2 by our observer. However, we think the robot is still trying to make other pedestrian as comfort as possible.

At the end of the video (starting from 13 minutes 48 seconds), we show 3 unscripted trials where we ask pedestrians to move at well and even try trapping the robot (trial 2, starting from 14 minutes 29 seconds). The robot is still capable of weaving through intensive interactions while navigating toward the given goal point which is on the left side of the road.

	A	B
#peds ≥ 2	67%	**67**%
#peds ≥ 4	64%	**75**%
#peds ≥ 6	58%	**75**%

Table 4.4: Controller A is a collision avoidance algorithm and controller B is our proposed model. With the number of pedestrians increasing, the difference between pure collision avoidance algorithm and socially compliant navigation algorithm becomes increasingly obvious.

4.2.2 A/B Testing

To demonstrate the ability of our approach to optimize both *naturalness* and *comfort*, we compare our approach against pure collision avoidance algorithm who only optimize the *comfort* aspect of socially compliant navigation. We design an A/B test experiment where we either use our approach or pure collision avoidance as controller to interact with subjects and ask them to identify whether they were interacting with socially compliant controller or not.

The experimental design was carried out on JMP 13 software by SAS Inc. The experimental plan was structured as a Main-and-two-factor-interaction plus quadratic, completely randomized D-optimal [12] design in 18 runs.

The inputs for the design were: Number of Pedestrians (2, 4, 6), Walking Direction (Parallel to the path of the oncoming robot, Parallel and Crossing the path, Crossing), and Controller (A, B). Pedestrians were entered as "discrete numeric" and the other two as "categorical". In "Parallel and Crossing" some pedestrians crossed the path of the robot (at approximately 2 meters distance) while others (at outer edge) continued on a parallel path. Pedestrians tried to walk steadily forward and at arms length from each other. The execution of the experiment was "blinded" in that the participants did not know which controller was active during any run.

For collision avoidance controller, we use the popular search based planning library (http://wiki.ros.org/sbpl). The results are summarized in Table 4.4. Controller A is a collision avoidance algorithm and controller B is our proposed model. Notice that in all runs, either with collision avoidance or our approach as controller, the robot never collides with a pedestrian and it always reaches the goal point within 0.5 meter tolerance. Looking all runs (with 2, 4, and 6 pedestrians), same amount of

subjects vote for controller A and B. While we increase the number of pedestrians to be considered, votes for collision avoidance controller decrease and votes for our approach increase. This highlights the significance of socially compliance in human-rich local interaction. With the number of pedestrians increasing, the difference between pure collision avoidance algorithm and socially compliant navigation algorithm becomes increasingly obvious.

Chapter 5

Conclusions and Future Directions

We present a data driven, deep generative approach for social navigation that covers both *comfort* and *naturalness* aspects. The proposed modeling of the intention force and social force as specially designed LSTMs makes the approach highly interpretable as an agent's intention can be conveniently visualized. We demonstrate the advantages of NaviGAN through an extensive set of experiments. The model is successfully deployed on a ClearPath Husky robot that allow us to conduct extensive studies in real-world robotic scenarios. As future directions for the research, we can:

1. Design human-robot interface that shows the intention, and modified social trajectory to convey *comfort* (let them feel safe) to other pedestrians that share the scene with the robot.
2. Incorporate obstacles for a complete navigation solution. With our current framework, we can extend our model to handle obstacles avoidance by simple coming up with a model to estimate obstacle force $\vec{F}_{obs}$ in social force model.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004. 1.1, 3.3

[2] Pieter Abbeel and Andrew Y Ng. Inverse reinforcement learning. In *Encyclopedia of machine learning*, pages 554–558. Springer, 2011. 1.1

[3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proc. CVPR*, pages 961–971, 2016. 1.2, 1.3, 2.3.1, 4.1.5, 3, 4.1.5

[4] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. Socially-aware large-scale crowd forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2203–2210, 2014. 1.2

[5] Gianluca Antonini, Michel Bierlaire, and Mats Weber. Discrete choice models of pedestrian walking behavior. *Transportation Research Part B: Methodological*, 40(8):667–687, 2006. 1.2

[6] Kai O Arras, Nicola Tomatis, and Roland Siegwart. Robox, a remarkable mobile robot for the real world. In *Experimental Robotics VIII*, pages 178–187. Springer, 2003. 1.1

[7] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial intelligence*, 114(1-2):3–55, 1999. 1.1

[8] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. *arXiv preprint arXiv:1809.08835*, 2018. 1.1, 1.3, 3.2.1, 4.1.5

[9] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1343–1350. IEEE, 2017. 1.1, 1.3, 3.2.2

[10] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 285–292. IEEE, 2017. 1.1

[11] Aurélie Clodic, Sara Fleury, Rachid Alami, Raja Chatila, Gérard Bailly, Ludovic Brethes, Maxime Cottret, Patrick Danes, Xavier Dollat, Frédéric Elisei, et al. Rackham: An interactive robot-guide. In *ROMAN 2006-the 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 502–509. IEEE, 2006. 1.1

[12] P Fernandes de Aguiar, B Bourguignon, MS Khots, DL Massart, and R Phan-Than-Luu. D-optimal designs. *Chemometrics and intelligent laboratory systems*, 30(2):199–210, 1995. 4.2.2

[13] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. *arXiv preprint arXiv:1805.01956*, 2018. 1.1

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1.3

[15] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number CONF, 2018. 1.2, 1.3, 4.1.2, 4.1.3, 4.1.5, 5

[16] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 1.3, 2.2.1

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1.3

[18] De-An Huang and Kris M Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *European Conference on Computer Vision*, pages 489–504. Springer, 2014. 1.2

[19] A. Jain, A. Roshan Zamir, S. Savarese, and A. Saxena. Structural-RNN: Deep Learning on Spatio-Temporal Graphs. *CoRR*, abs/1511.05298, 2015. URL http://arxiv.org/abs/1511.05298. 1.2

[20] Björn Jensen, Nicola Tomatis, Laetitia Mayor, Andrzej Drygajlo, and Roland Siegwart. Robots meet humans-interaction in public spaces. *IEEE Transactions on Industrial Electronics*, 52(6):1530–1546, 2005. 1.1

[21] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement

learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 1.1

[22] Diederik P Kingma and Jimmy Lei Ba. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*, 2014. 4.1.5

[23] 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. 1.1, 3.3

[24] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274, 2013. 1.1

[25] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *IJRR*, 35(11):1289–1307, 2016. 1.1, 3.3

[26] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726 – 1743, 2013. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2013.05.007. URL http://www.sciencedirect.com/science/article/pii/S0921889013001048. 1.1

[27] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3542–3549, 2014. 4.1.5

[28] Pinxin Long, Tingxiang Fanl, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018. 1.1

[29] Matthias Luber, Johannes A Stork, Gian Diego Tipaldi, and Kai O Arras. People tracking with human motion predictions from social forces. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 464–469. IEEE, 2010. 1.2

[30] Irene Macaluso, Edoardo Ardizzone, Antonio Chella, Massimo Cossentino, Antonio Gentile, R Gradino, Ignazio Infantino, Marilia Liotta, Riccardo Rizzo, and Giuseppe Scardino. Experiences with cicerobot, a museum guide cognitive robot. In *Congress of the Italian Association for Artificial Intelligence*, pages 474–482. Springer, 2005. 1.1

[31] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000. 1.1

[32] Illah R Nourbakhsh, Clayton Kunz, and Thomas Willeke. The mobot museum robot installations: A five year experiment. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 4, pages 3636–3641. IEEE, 2003. 1.1

[33] Billy Okal and Kai O Arras. Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2889–2895. IEEE, 2016. 1.1, 3.3

[34] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *European conference on computer vision*, pages 452–465. Springer, 2010. 4.1.5

[35] Mark Pfeiffer, Ulrich Schwesinger, Hannes Sommer, Enric Galceran, and Roland Siegwart. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2096–2101. IEEE, 2016. 1.1, 3.3

[36] Roland Philippsen and Roland Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *Proceedings. 2003 IEEE International Conference on Robotics and Automation, September 14-19, 2003, The Grand Hotel, Taipei, Taiwan*, volume 1, pages 446–451. IEEE Operations Center, 2003. 1.1

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 1.3

[38] Richard S Sutton. *Introduction to reinforcement learning*, volume 135. 1.1

[39] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 3.2

[40] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010. 1.1

[41] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hhnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972–999, 2000. doi: 10.1177/02783640022067922. URL https://doi.org/10.1177/02783640022067922. 1.1

[42] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803, Oct 2010. doi: 10.1109/IROS.2010.5654369. 1.1

[43] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal

n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011. 3.2.1

[44] Dizan Vasquez, Billy Okal, and Kai Arras. Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1341–1346, 2014. 1.1, 3.3

[45] A. Vemula, K. Muelling, and J. Oh. Social Attention: Modeling Attention in Human Crowds. In *ICRA*, 2018. 1.2, 1.3, 2.3.2, 4, 4.1.5

[46] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995. 4.2

[47] 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. 1.1, 3.3

[48] H. Zou, H. Su, S. Song, and J. Zhu. Understanding human behaviors in crowds by imitating the decision-making process. *arXiv:1801.08391*, 2018. 1.2