Learning Social Navigation from Demonstrations with Conditional Neural Processes

Yigit Yildirim and Emre Ugur
Computer Engineering Department, Bogazici University, Istanbul, Turkey

Sociability is essential for modern robots to increase their acceptability in human environments. Traditional techniques use manually engineered utility functions inspired by observing pedestrian behaviors to achieve social navigation. However, social aspects of navigation are diverse, changing across different types of environments, societies, and population densities, making it unrealistic to use hand-crafted techniques in each domain. This paper presents a data-driven navigation architecture that uses state-of-the-art neural architectures, namely Conditional Neural Processes, to learn global and local controllers of the mobile robot from observations. Additionally, we leverage a state-of-the-art, deep prediction mechanism to detect situations not similar to the trained ones, where reactive controllers step in to ensure safe navigation. Our results demonstrate that the proposed framework can successfully carry out navigation tasks regarding social norms in the data. Further, we showed that our system produces fewer personal-zone violations, causing less discomfort.

Keywords: social navigation, path planning, conditional neural process, data-driven control, random network distillation, generative adversarial networks, hybrid navigation architecture

Introduction

Researchers have been studying mobile robot navigation for decades. Many notable techniques have been proposed in this area over the years, such as (Burgard et al., 1999; Nourbakhsh, Kunz, & Willeke, 2003; Thrun et al., 2000), where safety and robustness features have been prioritized. In other words, the principal driving factor behind the development in this field has been collision avoidance (Fox, Burgard, & Thrun, 1997). On the other hand, as humans start to share their environment with robots, new requirements for mobile robot navigation have emerged.

Physical and mental aspects of safety were separately evaluated in Nonaka, Inoue, Arai, and Mae (2004). This separation reveals the need to question the psychological efficiency of the navigation systems of mobile robots. To ensure the smooth integration of robots into human environments, these systems must be social and as natural and understandable to humans as they are safe.

Kruse, Pandey, Alami, and Kirsch (2013) defines social navigation as navigating the robot in such a way that minimizes the annoyance that its motion produces. Efforts to decrease the annoyance or anxiety of the pedestrians interacting with a navigating robot can be included in the social navigation domain. To this end, the majority of the studies in the literature target to increase the comfortableness of the interaction. Fong, Nourbakhsh, and Dautenhahn (2003) asserts that people find it more comforting to interact with machines same the same way they interact with other people. Therefore, many researchers have aimed to decrease the discomfort that robot navigation generates by replicating human navigation with mobile robots.

The studies in the literature that aim to imitate human behavior in mobile robot navigation fall into two categories: manually coded controllers and learning-based ones. Manually coded controllers rely on hand-crafted optimization functions to resemble the motion of robots to that of humans. One of the notable studies of this category is the Social Force Model (SFM) (Helbing & Molnar, 1995). Based on behavioral techniques from the social sciences, SFM suggests that pedestrians move under the effect of specific abstract forces, just like particles in an electric field. While the navigational goal attracts the pedestrian, obstacles and other people exert repulsive forces. Despite its wide application (Farina, Fontanelli, Garulli, Giannitrapani, & Prattichizzo, 2017; Ferrer, Garrell, & Sanfeliu, 2013; Zanlungo, Ikeda, & Kanda, 2011), some researchers argue that hand-crafted models have limited applicability in controlled environments (Vasquez, Okal, & Arras, 2014) and that they are not general and applicable to different, varying social environments, especially during avoidance maneuvers (Kretzschmar, Spies, Sprunk, & Burgard, 2016). In real-world scenarios, social compliance of robot navigation requires adaptability. Kuderer, Kretzschmar, Sprunk, and Burgard (2012) proposed the use of data-driven approaches to create such adaptive controllers. Researchers have used numerous machine learning algorithms to create better adaptive, socially compliant navigation frameworks. One of the most popular algorithms is Inverse Reinforcement Learning (IRL) (Kim & Pineau, 2016;

Kitani, Ziebart, Bagnell, & Hebert, 2012; Kuderer et al., 2012; Vasquez et al., 2014). Given perfect expert demonstrations, IRL attempts to identify the underlying reward structure, which can be used by any Reinforcement Learning (RL) algorithm to create a human-aware navigation policy. The advantage of this approach is that the reward function is not manually determined but is a linear combination of a set of predefined features. However, the linearity assumption is considered a strong assumption in Wulfmeier, Ondruska, and Posner (2015).

Nonlinear rewards can better describe complex behaviors in many real-world problems (Levine, Popovic, & Koltun, 2011). Researchers have been using deep learning techniques to leverage this potential in social navigation. In Chen, Everett, Liu, and How (2017), Deep Reinforcement Learning was used to obtain a socially plausible navigation policy. As with other RL approaches, this procedure relies on a predefined reward. Similarly, Wulfmeier et al. (2015) extracted nonlinear rewards, assuming that the features shaping the reward function are known. On the other hand, Imitation Learning attempts to learn policies directly from the data, relaxing assumptions about the reward or its features. In Tai, Zhang, Liu, and Burgard (2018) and Gupta, Johnson, Fei-Fei, Savarese, and Alahi (2018), Generative Adversarial Networks were used for direct policy learning from demonstrations. These approaches provided advanced solutions to overcome the limitations mentioned above. However, these models required too much data for training (Che, Okamura, & Sadigh, 2020). On the other hand, learning from a small data set and generalizing to new configurations are desirable.

We also observe that most of the studies on the social navigation domain target only designing/learning the local controllers of the robots since they are responsible for producing motion commands. However, using only the local controller makes the robot vulnerable to the local minima problems (Koren & Borenstein, 1991), and might fail to navigate the robot to its target position. Today, typical robotic navigation systems adopt the two-layered hierarchical approach for path planning tasks (Orebäck & Christensen, 2003). A robot first calculates a trajectory in the so-called *global planning* phase given an environment. Then, the robot follows the computed trajectory with a controller in the so-called *local planning* phase.

This paper proposes a novel approach built on top of state-of-the-art neural network architecture, namely Conditional Neural Processes (CNPs) (Garnelo et al., 2018). Given multiple demonstrations of a task, CNPs can encode complex multi-modal trajectories. CNPs extract prior knowledge directly from training data by sampling observations and predicting a conditional distribution over any other target points. Taking advantage of these capabilities, we extended CNPs in two dimensions: to generate complete navigation trajectories in the global planning phase and to generate goal-directed behaviors while actively avoiding pedestrians in the local planning phase. At both levels, our approaches produce trajectories that show the characteristics of the demonstrated ones. They can learn complex, nonlinear, and temporal relationships associated with external parameters and goals. Like other neural network-based learning systems, our system may fail to generate trajectories or control signals when it faces very different situations from the experienced ones, i.e., when it is required to extrapolate to novel situations outside the training range. To detect and react to conditions that may lead to failure, we propose continuously monitoring the environment using a failure prediction system, detecting situations outside the training range, and falling back to a handcrafted reactive controller in case extrapolation is detected. We verified our system in a simulated mobile robot in different environments with static and moving pedestrians.

In the rest of this paper, we first give a literature review explaining the concepts used throughout this study. Then, we introduce our architecture in detail and elaborate on each system module. Later, we present our experiments and results. Finally, we conclude with a summary and future directions.

Related Work

Hybrid Path Planning

Traditionally, approaches to solving the path planning problem can be divided into two categories based on the environmental knowledge used: deliberate and reactive planning (Orebäck & Christensen, 2003). Deliberate planners use environmental knowledge through static maps and compute the robot's trajectory before execution. On the other hand, reactive planners rely on sensory information to deal with obstacles in a local frame around the robot. Both approaches have advantages and disadvantages. Affected by the hybrid deliberate/reactive paradigm (Dudek & Jenkin, 2010), a hybrid controller combining the two approaches has become a well-established approach to solving the path planning problem in recent years, (Murphy, 2019). In the following, we provide an overview of the two important building blocks of typical hybrid path planners, global and local planners, and the concept of social navigation.

Global Path Planning

In the first phase of standard hierarchical path planning, a global planning procedure is applied. On the static map of the environment, the task of a global planner is to create a path from the starting position to the destination. Global planners use utility functions to assign costs to the possible navigation trajectories they find. The use of utility functions allows the global planner to choose the trajectory with the desired properties, such as optimal length or time.

Prior to social navigation improvements, utility functions overlook the social aspects of the navigation trajectories of

robots. On the other hand, the trajectories with optimal physical properties may not be preferred from a social point of view. The utility function of a more socially competent global planner optimizes the trajectories with respect to the social norms that humans follow (Kruse et al., 2013).

Conventionally, many graph search algorithms have been applied to compute the trajectory between initial and target configurations, the most popular being A* explained in Kambhampati and Davis (1986). For a complete list of global planning approaches, see Giesbrecht (2004). Global planning itself is not sufficient to navigate the robot between two points. Local planning is required to create velocity commands appropriate for the case of new or dynamic obstacles.

Local Path Planning

The local planning procedures are used in the second phase of hierarchical path planning to realize the computed trajectories. The main objective of the local planner is to generate velocity commands to allow the robot to move between the checkpoints of the precomputed trajectory. In addition, it is the task of the local planner to avoid the obstacles near the robot using sensory information about the environment of the robot. Avoiding obstacles requires a reactive control paradigm since it is impossible to consciously plan for dynamic obstacles such as humans or other robots. There are many local planning algorithms in the literature, such as Borenstein, Koren, et al. (1991); Fox et al. (1997); Khatib (1985); Rosmann, Hoffmann, and Bertram (2015); Vadakkepat, Tan, and Ming-Liang (2000); Zhu, Yan, and Xing (2006). For a complete list, see Cai, Wang, Cheng, De Silva, and Meng (2020).

Many traditional local planners are well suited for this task from a safety point of view. On the other hand, the traditional controllers do not consider social norms despite providing physical safety. They view people as obstacles to be avoided. Even though creating social plans at the global planning level increases the social aspect of navigation, it is essential to implement social maneuvers when the robot encounters a pedestrian. Recent attempts to create local planners that take these norms into account paved the way for more socially compliant local controllers (Ferrer et al., 2013; Kim & Pineau, 2016; Kretzschmar et al., 2016; Vasquez et al., 2014).

Social Navigation

Social Navigation implies the exhibition of socially competent behaviors during the navigation of the robot. The nonverbal interaction caused by the navigation of the robot may be improved by the integration of social and cultural norms that people follow. Kruse et al. (2013) describe the benefits of social navigation as follows: it increases the comfort of the people around the robot, improves the naturalness of the robotic platform, and enhances the sociability of the robot. The concept of social navigation lies in the intersection of two fields: navigation and human-robot interaction. Figure 1 presents a comparison of the purely safe approach with a socially compliant version. A robot with a perfectly safe navigation plan might disregard the importance of the comfort level of the encountered pedestrians, such as the one on the left. In contrast, although non-optimal, the executed navigation trajectory on the right is more socially compliant since it cares about the comfort level of the people around.

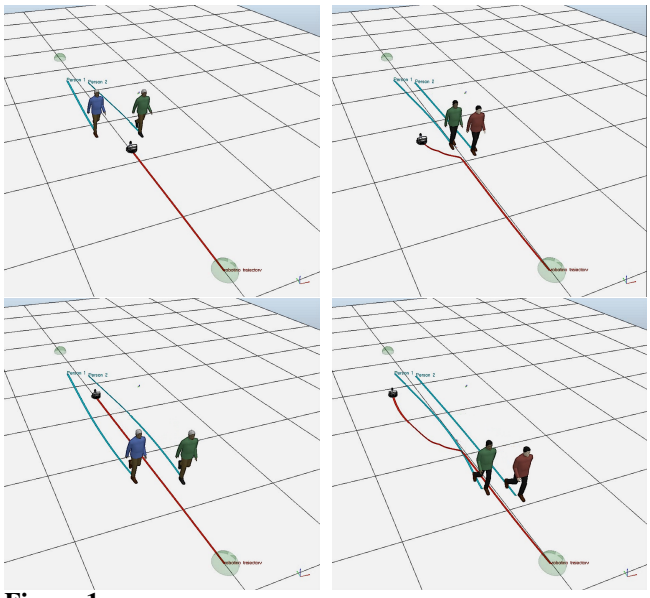


Figure 1

Comparison between the regular and social navigation. On the left, the robot passes between a group of two pedestrians, taking an energy-efficient trajectory. This behavior has the disadvantage of disturbing the people encountered on the way. The navigation trajectory on the right prioritizes the people's comfort instead of efficiency. Therefore, it is less disturbing and expected from socially compliant robots.

The early works in the domain are influenced by studies in social sciences. The concept of proxemics, introduced in Hall (1966), defines abstract social zones around the people and provides a basis for many studies in socially-compliant robot navigation (Asghari Oskoei, Walters, & Dautenhahn, 2010; Huang, Li, & Fu, 2010; Lam, Chou, Chang, & Fu, 2010; Mead, Atrash, & Matarić, 2011; Syrdal, Koay, Walters, & Dautenhahn, 2007). Although these pioneer studies relied on the predefined set of rules to achieve social navigation, they drew attention to the subject and made it more popular.

Another important study in social sciences is the Helbing and Molnar (1995) where researchers introduce the Social Force Model (SFM) to explain the navigational behaviors of

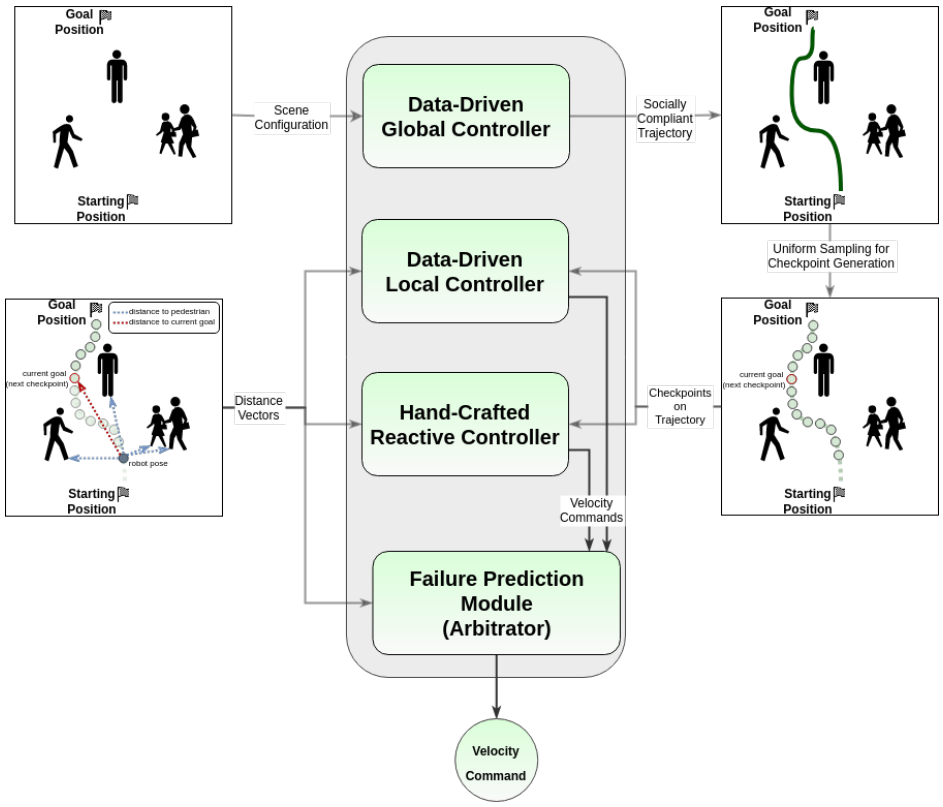


Figure 2

The overview of the proposed navigation pipeline. The Data-Driven Navigation System is composed of four modules: Data-Driven Global Controller, Data-Driven Local Controller, Failure Prediction, and Hand-Crafted Reactive Controller. When a navigation task is given to the robot, the Data-Driven Global Controller generates a continuous navigation trajectory. On this trajectory, via-points are created, which must be reached one by one by the Data-Driven Local Controller. These modules are data-driven and are prone to extrapolation errors. Such cases are detected by the Failure Prediction Module, which transfers the control of the mobile robot to the Hand-Crafted Reactive Controller temporarily.

humans. They define a set of functions to calculate the local movements of pedestrians to reach a global goal. The simplicity of the SFM model causes many researchers in robotics to adopt the approach to move the robots as humans do, (Farina et al., 2017; Ferrer et al., 2013; Zanlungo et al., 2011).

As of late, data-driven approaches have become more prevalent in explaining human behavior since they are more adaptable to many situations. Many researchers apply Inverse Reinforcement Learning (IRL) to calculate a reward function that describes the navigational behavior of the human, (Kim & Pineau, 2016; Kitani et al., 2012; Vasquez et al., 2014). Another stream of work uses Deep Reinforcement Learning to generate human-aware robot navigation, (Chen et al., 2017). These studies assume the availability of either the reward or the features that compose the reward. To relax this assumption, approaches that learn the social norms merely from the data became more popular.

In Alahi et al. (2016), researchers use networks with Long-Short Term Memory cells (LSTMs). Later, an improved version of this study is presented in Gupta et al. (2018), where researchers use LSTMs inside a generative adversarial setting. These studies successfully predict human navigation trajectory in a limited local frame. Despite being considerably close to the social navigation domain, the trajectory prediction methods in these studies cannot be directly applied to mobile robots. A robot placed in a real-world environment has to meet serious time-complexity considerations. Moreover, these approaches are limited to local frames, while path-planning on mobile robots requires a global navigation goal (Latombe, 1991).

Data-driven approaches generally process navigation trajectories in datasets to realize social navigation. However, as mentioned in Mavrogiannis et al. (2021), the scarcity of established datasets has been a significant issue in the domain. Until recently, many studies in the literature (e.g. Trautman and Krause (2010); Vemula, Muelling, and Oh (2018))

have been using pedestrian datasets to train models, such as Lerner, Chrysanthou, and Lischinski (2007) and Pellegrini, Ess, Schindler, and Van Gool (2009). Although such datasets provide real-world interactions among pedestrians, they do not contain human-robot interaction. Therefore, pedestrian datasets may overlook any possible effect of an embodied robotic agent's presence on a neighboring pedestrian's navigation trajectory. Lately, we have witnessed an increase in datasets focusing on human-robot interactions for social navigation studies. Manso, Nuñez, Calderita, Faria, and Bachiller (2020) aims to evaluate the comfort levels of pedestrians around a robot in an environment. On the other hand, this dataset does not capture the navigation dynamics of the robot and pedestrians as it merely focuses on static scenes. Recent datasets, such as Yan, Duckett, and Bellotto (2017), Martin-Martin et al. (2021), Karnan et al. (2022), address both shortcomings, where researchers control robots by teleoperation in real-world environments to record navigation demonstrations.

Our Method

This work proposes a hybrid data-driven navigation system that uses advanced neural techniques in global and local layers coupled with a novel execution monitoring module for fault-tolerant intelligent navigation. Figure 2 demonstrates the overview of our proposed system. Our navigation system consists of four modules: Data-Driven Global Controller, Data-Driven Local Controller, Failure Prediction, and Hand-Crafted Reactive Controller. After introducing these modules in this section, the following subsections will provide the details. First of all, given a target position, the Data-Driven Global Controller is responsible for producing a complete navigation trajectory consistent with the demonstration trajectories previously provided to the robot. Next, several via-points are uniformly sampled from the generated navigation trajectory. The second module, namely the Data-Driven Local Controller, is responsible for generating motion commands to reach each via-point one by one. While maneuvering the robot, this controller reacts to the dynamic changes in the vicinity of the robot and changes the target to the next via-point as soon as the current target is reached. This procedure continues until the robot reaches the final target, i.e., the given goal position. Suppose the local controller fails to follow the trajectory at any point; a pedestrian may be blocking the path, for example. In that case, the global controller can be called again to recalculate a new trajectory on the same navigation task. The first two modules use data-driven approaches, which learn from provided trajectories, and can be used to learn social competencies for social navigation. The proposed navigation system achieves learning and control in these modules using a neural network family. Similar to other neural network architectures, while models can interpolate to novel situations within the training range, the learned models are prone to extrapolation errors when they run on inputs outside their training range. If unattended, such errors can cause collisions with pedestrians during navigation. Two additional modules are incorporated into our system to detect and react to the extrapolation cases before any collision or failure happens: the Failure Prediction Module and the Hand-Crafted Reactive Controller. The Failure Prediction Module observes the entire operation of the system and is responsible for detecting outlier situations. If this module detects that data-driven controllers are queried with an input outside their training range, the control is temporarily transferred to the Hand-Crafted Reactive Controller, ensuring the safety of navigation. When the risk of failure vanishes, i.e., the robot and its environment are detected to be in the training range again, the data-driven modules take back the control of the robot. These components are combined to form a full-fledged navigation pipeline. In the following, we explain these modules in detail.

I - Data-Driven Global Controller

In our work, the task of global planning is to generate a complete trajectory for navigation. A set of discrete via-points can be sampled from this trajectory to enable the local planner to reach each via-point successively. The Data-Driven Global Controller aims to learn a parametric distribution of social navigation trajectories that reflect the social norms of the people. Therefore, after learning the underlying distribution of socially acceptable trajectories, this module can reactively change the navigation trajectory according to the configuration of the environment.

Training the Data-Driven Global Controller: We recorded a set of expert demonstrations to teach the trajectory generation function to our model. Each demonstration contains an environment configuration and a navigation trajectory close enough to the optimal social behavior in the corresponding environment. The environment configuration is the concatenation of the following three elements: the robot's start and goal positions and the pedestrians' positions. In addition, navigation trajectories are stored as a set of *(x, y)* coordinates with corresponding time steps.

Formally, the set of all trajectories is a collection of N expert demonstrations, denoted by $D = \{\tau^i\}_{i=1}^{N}$. Each navigation trajectory τ is a set of 2D positions at each time step; $\tau = \{(x_j, y_j)\}_{j=1}^{T}$. Furthermore, the configuration C of each scene is formed as $C = \{(x_{start}, y_{start}), (x_{goal}, y_{goal}), ((x_{p_1}, y_{p_1}) \dots (x_{p_K}, y_{p_K}))\}$, where (x_{p_k}, y_{p_k}) represents the 2D position of one of the K pedestrians in the scene.

The overall training procedure is depicted in Figure 3. At training time, the system uniformly samples a navigation trajectory τ^i from D. Afterward, n observation points are uniformly sampled on this trajectory, where n is a random number between 0 and n_{max}. The navigation trajectory is the

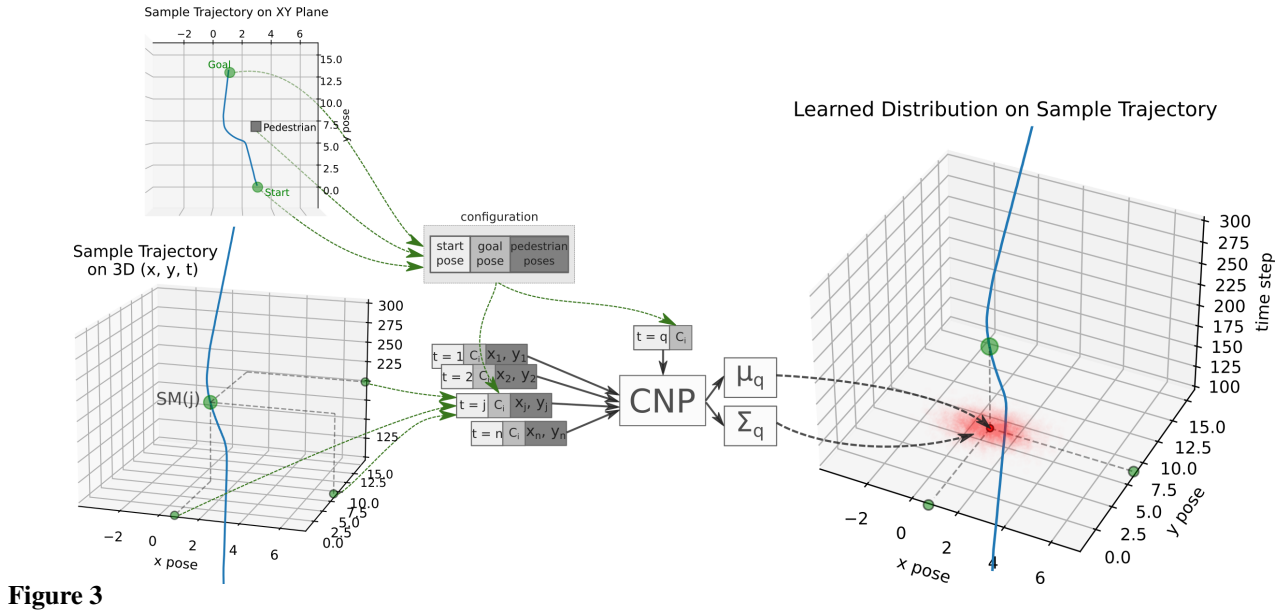


Figure 3

Training the Data-Driven Global Controller with a demonstration trajectory. The configuration of the environment and the demonstration trajectory on this environment are gathered and processed. Later, this data is given as input to the CNP structure. The model predicts a bivariate normal distribution of 2D positions on the XY plane for a queried time step. The log-likelihood of the actual data point given the predicted distribution produces the loss that is back-propagated to tune the weights of the neural network structure.

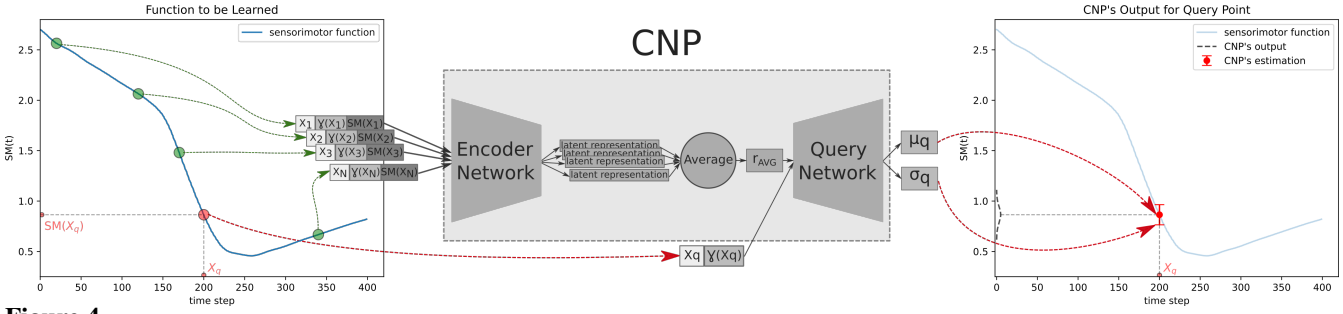


Figure 4

The general layout of the training phase of our model. First, a random number of observation points are sampled randomly on SM() trajectory. An observation point consists of the input (X), the task parameter for that input ($\gamma(X)$), and the value of the function for that input ($SM(X)$). Observation points are fed to the Encoder Network. Obtained latent representations are averaged to retrieve a compact representation of the trajectory. The query network is run with a random query point to produce the prediction of the system. This prediction causes a loss that is back-propagated through both networks.

sensorimotor function that the model learns in our representation. It is a function of time and is shaped by the scene's configuration. The realizations of this function on different time steps correspond to 2D positions in the real world. That is, $\tau = \{SM(t)\}_{t=1}^{T}$, where $SM(t) = (x_t, y_t)$. Moreover, we represent the configuration, C, of the environment with the task parameter, using the model's γ function. Hence, in this representation, $\gamma(t) = C$.

Figure 4 introduces the underlying neural network model with the corresponding Encoder-Decoder structure. The Encoder Network takes in $(t, \gamma(t)$ and $SM(t))$ tuples, produces their corresponding latent representations, and applies averaging operation to generate a general representation to encode the n tuples. On the other hand, the Query Network is responsible for generating the distribution related to the position of the robot $(SM(t_q))$ at any time point t' where the gen-

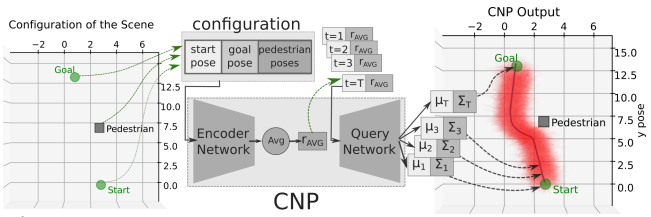


Figure 5

Generating an entire navigation trajectory for the configuration given on the left panel. This configuration is given as input to the network, as shown in the middle column. The predicted trajectory is shown on the right. Note that the solid line corresponds to the sequence of the mean positions and the red shaded area illustrates the variance information. For details, please refer to the text.

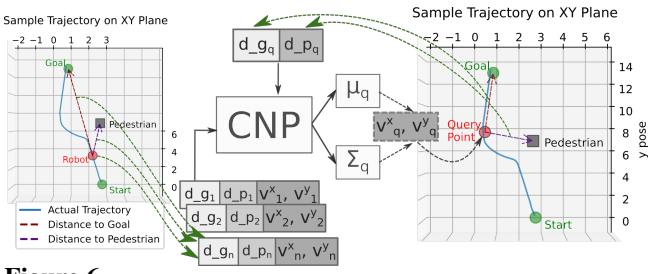


Figure 6

Training the Data-Driven Local Controller with a demonstration trajectory. Along a demonstration trajectory, several observations are sampled randomly. Each observation contains the relative position of the goal, d_g, relative positions of the pedestrians, d_p, and the velocity command, v. The velocity prediction of the model for the query point provides the loss, which is back-propagated through both networks of the CNP structure.

erated position is required to be consistent with the average latent representation of n tuples. Therefore, given the average latent representation of n ($t, \gamma(t)$ and $SM(t)$) tuples and a random target time point t_q, the Query Network produces the distribution of the predicted position of the robot, i.e., outputs a bivariate normal distribution with parameters (μ_q, Σ_q). With this output, the complete neural network model is trained using the following loss function:

$$\mathcal{L} = -\log P(\, SM(t_q) \,|\, \mu_q,\ softmax(\Sigma_q)\,) \qquad (1)$$

where SM(t_q) corresponds to the actual (x, y) coordinates of the trajectory at time t_q.

Querying the Trained Data-Driven Global Controller: After the training of the Data-Driven Global Controller with given navigation trajectories, it can be queried to generate new navigation trajectories given new environment configurations, as illustrated in Figure 5. The underlying system can predict the position of the robot at any time point. Therefore, the time step is used as the input, while the scene configuration is used as the task parameter to generate the corresponding position of the desired trajectory. The Data-Driven Global Controller can be queried simultaneously and independently for all time points. In the end, this module outputs an entire trajectory of 2D coordinates - from the starting position to the goal position- as a function of time. When trained with social navigation trajectories, the learned function is a social trajectory generator that avoids pedestrians in the intended course of the navigation.

II - Data-Driven Local Controller

Typically, the main task of a local planner is to move the robot through the via-points on the navigation trajectory computed by the global planner. Moreover, it is the task of the local planner to navigate the robot without colliding with pedestrians. In our system, the Data-Driven Local Controller is used to accomplish both tasks and preserve the characteristics present in the demonstrations.

This module is built on top of a CNP-based neural architecture. As in traditional local planners, this module generates goal-directed behavior by targeting short-distance via-points as intermediate goals. It uses the relative position of the next checkpoint as input, X. To achieve collision-free navigation, relative positions of pedestrians are passed to the network as a task parameter, $\gamma(X)$. This gives the Data-Driven Local Controller the ability to reactively adjust its output with respect to the changing pedestrian positions.

In addition to the basic tasks, the neural network within the Data-Driven Local Controller can learn the social characteristics present in the local evasive maneuvers of the people, using only the data from their navigation trajectories. In the end, the Data-Driven Local Controller decides which action to take in the form of velocity commands of the mobile base.

Training the Data-Driven Local Controller: The local planners use sensory information to understand their local frames. Our system processes the low-level sensory information to obtain high-level parameters, such as relative position of pedestrians and relative position of the goal position. The Data-Driven Local Controller reactively responds to the changes in these high-level parameters. More importantly, it does so while preserving the characteristics present in the previously learned demonstrations. Therefore, when the module is trained with socially acceptable trajectories, it can learn social norms. The demonstration trajectories are recorded as a set of tuples $(d_{goal}, [d_{pedestrian}], v)$, where d_{goal} is a 2D vector of the relative position of the current goal, $[d_{pedestrian}]$ is an array of 2D vectors representing relative po-

sition of the pedestrians in the scene, and v is a 2D vector representing the velocity of the base on the 2D plane.

Prior to the training, trajectories are processed and converted into a set of observations that the neural network of the Data-Driven Local Controller uses. An observation is formed by the concatenation of three parts: $X, \gamma(X)$ and $SM(X, \gamma(X))$, where $X = d_{goal}$, $\gamma(X) = [d_{pedestrian}]$, and $SM(X, \gamma(X)) = (v^x, v^y)$, and v^x, v^y correspond to X and Y components of the velocity vector. Figure 6 illustrates the training procedure. The training process uniformly samples a trajectory τ^i from the set of demonstrations, D. n observations are uniformly sampled, where n is a random number between 0 and n_{max}. These observations are fed into the *Encoder Network* to obtain latent representations, which, in turn, are averaged to create the average latent representation of the entire trajectory. This is given as input to the *Query Network*, along with a random query point X_q, and the value of the task parameter for this query point, $\gamma(X_q)$. The *Query Network* outputs a bivariate normal distribution with parameters (μ_q, Σ_q) which represents the normal distribution that describes the model's prediction about the velocity command. The loss calculation is the same as in Equation 1, which is back-propagated through the entire CNP structure.

Note that in the previously described Data-Driven Global Controller, latent representation encoded the entire trajectory for the corresponding environment and the navigation task and was conditioned with successive time points (independently) to generate the successive points of the trajectory that the robot is supposed to follow. In the Data-Driven Local Controller, on the other hand, the latent representation encodes instantaneous control command rather than the entire trajectory. It, therefore, is conditioned on the relative position of the checkpoint to generate the corresponding velocity command.

Querying the Trained Data-Driven Local Controller: After training the encoder and the query networks, the model can be run for specific scene configurations. Conditioned on the start and goal positions, the use of the task parameter $\gamma(X)$ gives the model the ability to reactively change the velocity commands with respect to changing pedestrian positions and possibly changing checkpoint positions in case of the re-computation of the trajectory. At any time, the relative position of the robot with respect to the current goal (d_{goal}) and the closest pedestrian ($d_{pedestrian}$) can be concatenated and given to the model as input. The neural network outputs the velocity command (v^x, v^y) to be executed by the robot. At test time, the query is fast enough to be used in real-time applications. Therefore, this module can be used as a local controller that generates velocity commands following the characteristics of the demonstrations, which is used for social navigation.

III - Failure Prediction Module

The previously described Data-Driven Global and Local Controllers learn representations inside the domain they are trained. Therefore, these modules are inherently prone to extrapolation cases when queried outside the learning range. These cases may lead to anomalous behavior and, in practice, generates undesirable velocity commands. Although infrequent, such anomalies may cause collisions and are unacceptable in navigation systems where the safety of pedestrians is of the utmost importance.

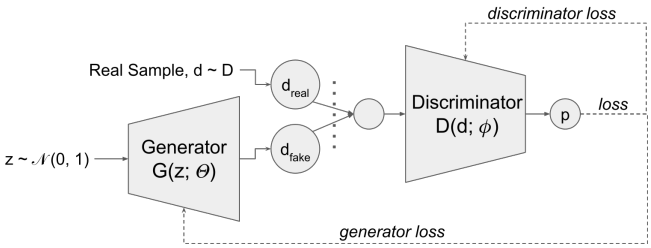
To avoid such undesired behavior, we provide the navigation system with the capability to continuously observe the states it encounters during a navigation task for their likelihood of lying outside its training domain. Firstly, we leverage the ability of Generative Adversarial Networks (GAN) (I. J. Goodfellow et al., 2014) to learn the input-space distribution. There are many variants of GANs in the literature, such as Wasserstein GAN (W-GAN) (Arjovsky, Chintala, & Bottou, 2017). For our case, standard GAN worked more stably compared to the W-GAN.

Figure 7a depicts the training procedure. On the same dataset that the local CNP is trained, we train a GAN. GAN takes as input $[d_{pedestrian}]$ from actual trajectories from the dataset. The generator network attempts to produce realistic candidates to deceive the discriminator network in deciding whether its input belongs to the actual dataset or not. The objective function, given in Equation 2, follows the standard minimax loss introduced in I. J. Goodfellow et al. (2014).

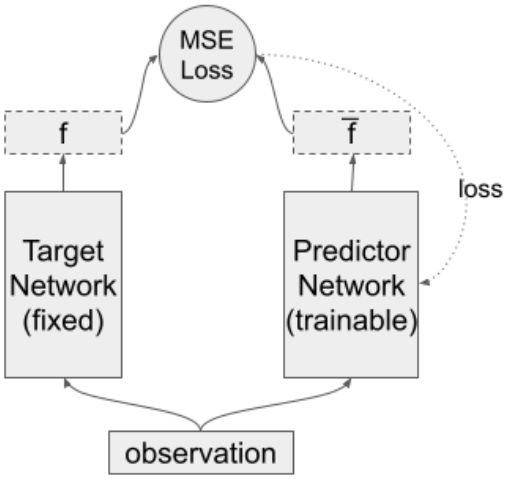
$$\min_G \max_D \mathbb{E}_d \left[\log D(d)\right] + \mathbb{E}_z \left[\log \left(1 - D(G(z))\right)\right] \quad (2)$$

During the training of the networks with their corresponding losses, the generator network becomes better at producing realistic candidates, and the discriminator network becomes better at discriminating against them. After the training, the discriminator learns the distribution that the actual data belongs to. We use the output of the discriminator network, which is the probability of an encountered state being sampled from the same distribution as the training data, in predicting the extrapolation.

Although the system benefits from using GANs, these architectures are often criticized for being unstable due to oscillation and mode collapse, I. Goodfellow (2016). Therefore, we have implemented another approach, named Random Network Distillation (RND) (Burda, Edwards, Storkey, & Klimov, 2018) in this module for detecting out-of-distribution (OOD) samples. This approach uses two subnetworks; the first one - the target network - is a fixed and randomly-initialized multi-layer perceptron (MLP) and the second - the predictor network - is a standard fully connected MLP. The approach evaluates the novelty of an encountered system by calculating the distance of the output that two subnetworks produce. Initially, the two would produce different



(a) *GAN training for predicting the extrapolation. While the actual data is sampled from demonstration trajectories, the generator network produces realistic candidates. The discriminator network learns the distribution that the navigation data belongs. As a result, it learns to attribute small probabilities to cases with unusual pedestrian positions.*



(b) *RND training for predicting the extrapolation. The observation is sampled from demonstration trajectories. The target network outputs random values, and the predictor network learns to mimic them. After training, two networks output similar values for familiar observations.*

Figure 7

Approaches used in the Failure Prediction Module. The RND approach is preferred in this study due to its stability.

results. By training the predictor, it starts to output similar values as the target network does. Upon training, when an OOD input is queried, two sub-networks output very different values, producing a high error value and enabling our system to predict extrapolation errors before they occur. The structure of this architecture is given in Figure 7b.

The model is trained with the relative positions of pedestrians and the goal at each time step. This information is extracted from the same expert demonstrations that the data-driven controller is trained with. After training the RND model, the distance between its sub-networks is used to detect whether a new data point is from the training range.

IV - Hand-Crafted Reactive Controller

Whenever the Failure Prediction Module outputs a small probability for an observation point, the navigation system concludes that it is about to extrapolate. This usually leads to erratic movements and potentially a collision with a pedestrian. To control the robot safely in those situations, we created the Hand-Crafted Reactive Controller. This module implements the Social Force Model (SFM) to move the robot in a safe and socially compliant manner. On the other hand, the SFM is not flexible enough as many real-world applications require since the attractive and repulsive components are defined and tuned manually. Therefore, the pipeline uses the Hand-Crafted Reactive Controller as a fallback module. During the period it controls the robot, the Failure Prediction Module continues to check whether it is safe for the Data-Driven Local Controller to take back control.

Experiments and Results

Our system was verified in the CoppeliaSim simulation environment (Rohmer, Singh, & Freese, 2013) that includes an omnidirectional robot platform, namely Robotino (Festo Robotics, 2020). For details about the dataset used in training all deep models, including CNPs, GAN and RND, please refer to Appendix A. Also, hyperparameters used in training all networks are given in B.

Analysis of the Generated Global Trajectories

The application of neural networks in global planning tasks is an established procedure. Many studies in the litera-

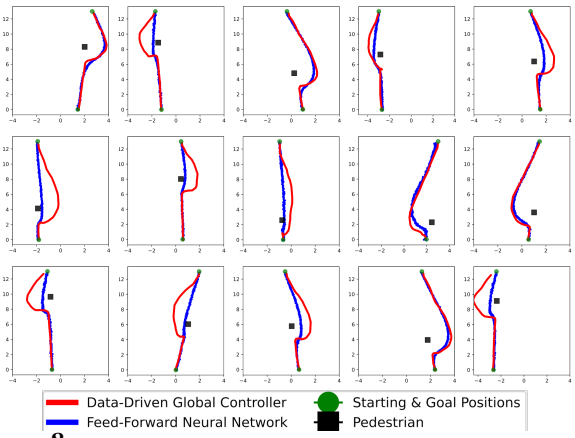


Figure 8

The comparison of trajectories generated by the Data-Driven Global Controller and a feed-forward neural network on several environment configurations. Results show that the Data-Driven Global Controller produces less disturbing trajectories for pedestrians.

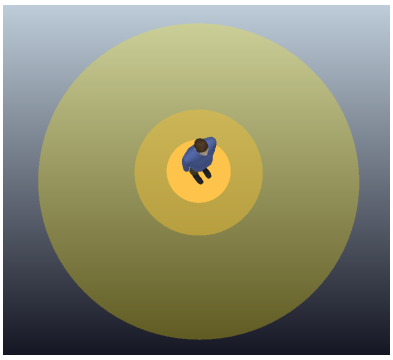


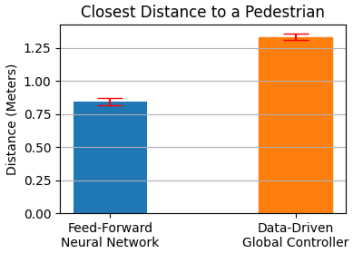
Figure 9

Illustration of the proxemics zones. The theory of proxemics defines four notional zones surrounding a human that influence the behavior of the individual by affecting the type and the content of the interaction. These zones (from inner to outer) are Intimate (0-0.5 m.), Personal (0.5-1 m.), Social (1-4 m.), and Public (4-8 m).

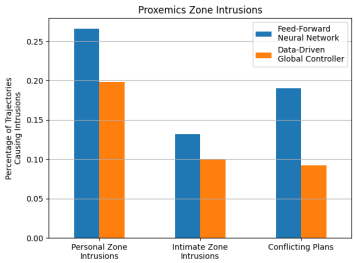
ture propose variants of neural networks that optimize for the shortest or the minimum-energy trajectories (Glasius, Komoda, & Gielen, 1995). In this part, we aim to emphasize the capability of our structure over the standard neural network approach from the perspective of social navigation.

On the dataset of trajectories described in Appendix A, we trained our Data-Driven Global Controller and a standard feed-forward neural network (FFNN) with five layers for comparison. After training, both networks were queried in novel environments with different starting and goal positions and with different pedestrian positions. A subset of the trajectories generated by both networks is given in Figure 8. In most cases, both approaches could generate trajectories that allow the robot to avoid pedestrians, which are shown with black dots. In many cases, though, the FFNN failed to generate global paths that avoid the pedestrian in a socially compliant manner, whereas our system could. We believe that the difference in the performance was due to the inability of standard FFNNs to encode multiple modes of operations, in other words, multiple trajectories for the same or very similar environments. In detail, given multiple trajectories that avoid the same pedestrian from different sides, our system can encode different modes in its robust latent space. In contrast, standard FFNNs learn an average representation from multiple trajectories in the same environment. Therefore, the produced trajectory by FFNNs corresponded to the average trajectory of the demonstrated ones.

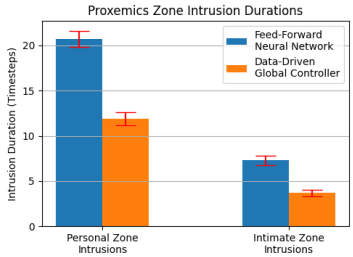
In order to further analyze and compare these approaches from the social navigation perspective, we used the metrical comparison on the concept of proxemics. Defined by Hall (1966), the theory of proxemics investigates the spatial aspects of nonverbal communication. It suggests that interpersonal spaces are critical determinants of social interactions. In addition, there are four abstract zones surrounding a human. An illustration of them is given in Figure 9. The physical intrusion of these zones is only allowed under certain circumstances; for example, people can comfortably allow their family and close friends in their personal zones. On the contrary, disallowed intrusion of these zones usually leads to discomfort or anxiety.



(a)



(b)



(c)

Figure 10

Comparison of approaches in terms of distance-based metrics. Figure 10a shows the closest distances to a pedestrian both approaches produce. Figures 10b, 10c show counts and durations of zone intrusions. Trajectories generated by the Data-Driven Global Controller pass close to the pedestrian less frequently, causing less discomfort. Proxemics-zone intrusions support this argument. Refer to the text for more details.

We present the metrical comparison of these approaches

in Figure 10. Using 1000 randomly generated environments, we first measure the closest distance of the generated trajectories to a pedestrian in the environment and compare the methods based on the closest distances. Figure 10a shows the comparison. The Data-Driven Global Controller is better at producing trajectories that do not come too close to pedestrians. The closer the distance to a pedestrian, the more disturbing the robot becomes (Tipaldi & Arras, 2011).

Moreover, Vasquez et al. (2014) uses a distance-based comfort metric to compare navigation approaches from the social navigation perspective: the number of intrusions of the intimate and personal zones. In addition to that, we also use the duration of zone intrusions as an indicator of discomfort. A more extended period of intrusion of proxemics zones naturally leads to a higher level of anxiety. As shown in Figures 10b and 10c, our approach performs substantially better than the standard FFNN approach in all cases. The differences between the two approaches in both closest-distance and zone-intrusion values are statistically significant upon the application of the t-test and z-test with a confidence level of 0.95.

Analysis of the Local Controller: Evasive Maneuvers

Our model makes instantaneous action decisions at the local-controller level to respond to the changes in the environment. For convenience, the model considers only the closest pedestrian, but the underlying neural network can handle multiple pedestrians. When it is inevitable to diverge from a straight path due to a possible conflict with a pedestrian, the local controller navigates the robot safely, following the norms taught by the expert demonstrations. To establish these claims, the performance of the trained local controller is assessed on a set of simulated tasks.

Firstly, the obstacle avoidance capability of the Data-Driven Local Controller was tested in configurations with a moving pedestrian. Although the training dataset does not contain any scenarios with moving pedestrians, our local controller could generalize to such cases. In Figure 11, the robot alters its trajectory, shown in blue color, dynamically to steer away from the pedestrian, whose trajectory is shown in red color. The execution of this motion prior is more apparent in Figure 12. In this case, the robot maneuvers multiple times to avoid any probable zone intrusions while moving toward its goal.

Second, we tested the Data-Driven Local Controller in a scenario where multiple pedestrians were situated. Although the controller is trained in environments with single and stationary pedestrians, it can scale this behavior into more crowded environments by considering only the closest pedestrian at test time. In Figure 13, the social aspect of the controller is rather apparent. Instead of going towards the goal directly, which is efficient and physically safe, the robot exhibits a social behavior of steering away from the pedestrians as its priority. Since such a norm was present in the expert behavior, the Data-Driven Local Controller captures this as a motion primitive.

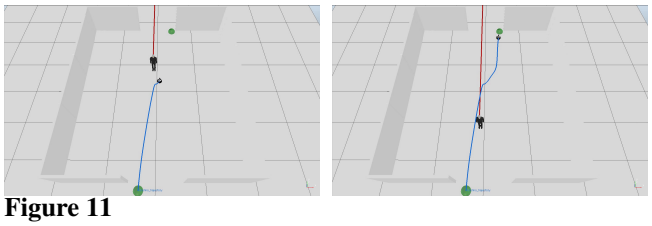
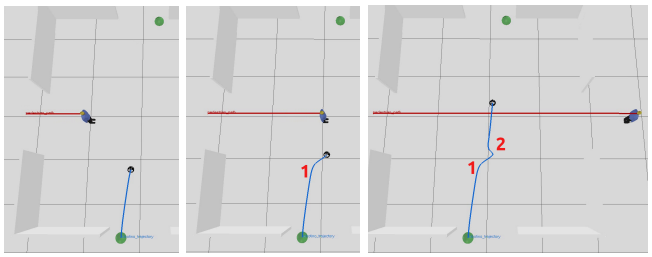


Figure 11

Snapshots from the scene where the pedestrian moves on a vertical trajectory shown in red. The robot, controlled by the Data-Driven Local Controller, aims to reach its target, shown at the top. When it encounters the pedestrian, it evades the pedestrian, reactively changing its trajectory, shown in blue.



(a) (b) (c)

Figure 12

Snapshots from the scene where the pedestrian is moving on a horizontal trajectory, shown in red whereas the trajectory of the robot is shown in blue. Figure 12a shows that the robot initially aims to reach its target, shown at the top with the green sphere. Figure 12b shows that it alters its trajectory first to evade the pedestrian on its left. As the pedestrian continues to move, Figure 12c shows that the robot again makes a maneuver to evade the pedestrian on its right.

Comparison of Local Controllers

To assess the success of our system, we devised a set of 25 simulated tests with three local controllers; Social Force Model, CNP trained on Simulation, and CNP trained on SCAND. SCAND (Karnan et al., 2022) is a real-world dataset gathered on indoor and outdoor social environments with human demonstrators. More details on the dataset are given in Appendix A. The quantitative comparison is given in Table 1.

Several quantitative metrics are used to compare the three controllers. Average Displacement Error (ADE) and Average Acceleration (AA) metrics are proposed in Mavrogiannis et

Figure 13

Snapshots from the scene where the robot is passing through several pedestrians. Instead of going directly to its target, the robot takes a longer trajectory (shown in blue), complying with the social norms present in the dataset. This behavior decreases proxemics-zone intrusions.

	Social Force Model	CNP on Simulation	CNP on SCAND
ADE	0.43 ± 0.01	0.53 ± 0.01	0.72 ± 0.02
PL	13.18 ± 0.01	13.49 ± 0.03	13.10 ± 0.01
ATG	22.83 ± 0.08	34.83 ± 0.38	69.86 ± 0.80
AA	5.24 ± 1.51	6.51 ± 0.07	-0.06 ± 0.07
PZIC	0	0	3

Table 1

Three local controllers are compared in simulated tests on 25 environments on five dimensions. Average Displacement Error (ADE) shows how much the robot steers away from the straight line from the start position to the goal position. Path Length (PL) is used to measure the length of the generated trajectories whereas Average Time to Goal (ATG) shows how long it takes for the robot to get to the goal position. Average Acceleration (AA) is reported by summing up the changes in the acceleration throughout the navigation task. Proxemics Zone Intrusion Counts (PZIC) measure the number of personal zone intrusions. SI units are used in the comparison. Refer to the text for details.

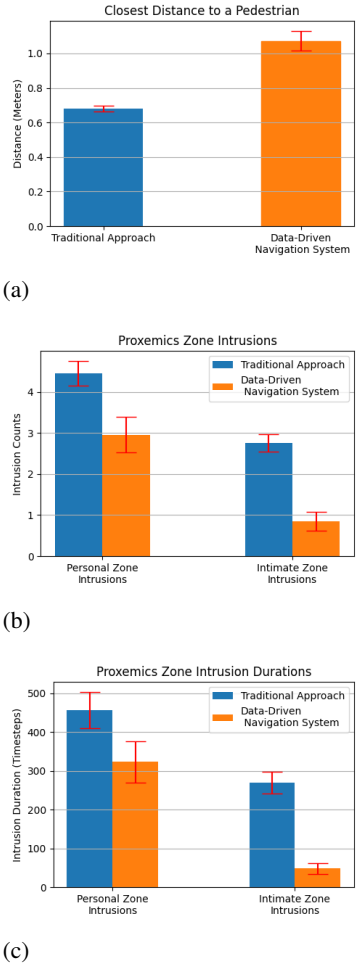
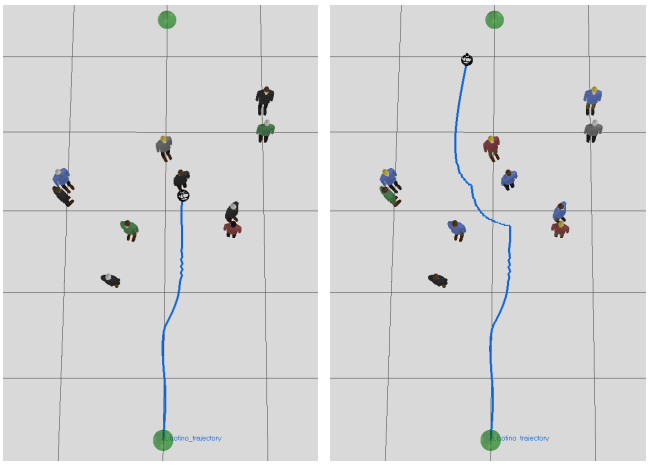


Figure 14

Comparison of approaches in terms of distance-based comfort metrics. In Figure 14a, we show the closest distances to a pedestrian both approaches produce. Our pipeline minimizes the discomfort by staying distant from the pedestrians during navigation. This characteristic is also reflected in Figures 14b and 14c where we compare two methods in terms of zone-intrusion metrics. The count and the duration of zone intrusions are significantly less for our system. Refer to the text for more details.

al. (2021); whereas Path Length (PL) and Average Time to Goal (ATG) in Okal and Arras (2016). Proxemics Zone Intrusion Counts (PZIC) metric is proposed in Vasquez et al. (2014). According to these tests, the most notable difference is the velocity each agent prefers in the same condition. Demonstrators seem to prefer lower speeds around people, whereas our simulated controller and the one trained on this data travel at higher speeds. Therefore, even though PL values are very close, the controller trained on real-world data scores substantially higher ATG values. This situation is also reflected in the AA dimension. The controller trained on real-world data tends to change its speed less frequently than the other two. On the other hand, this controller makes more personal zone intrusions, signaling that human demonstrators prefer slowly diverging from other pedestrians' paths in case of encounters.



(a) (b)

Figure 15

Navigation system with and without the Failure Prediction Module in the same environment. Figure 15a shows when the robot relies only on the data-driven modules of the navigation system. This leads to an extrapolation error and eventually a collision. In Figure 15b, the Failure Prediction Module detects this situation and gives the control temporarily to the Hand-Crafted Reactive Controller, enabling the robot to move safely within the cluttered area.

Performance of the Complete System

Finally, we demonstrate the contribution of our pipeline in terms of social navigation with a quantitative comparison. In the same environment as shown in Figure 13, we randomly distributed the pedestrians and compared the executed navigation trajectories of the proposed system and a traditional navigation system. The traditional planner implements the A* algorithm at the global level (Kambhampati & Davis, 1986), and the Elastic Bands approach at the local level (Quinlan & Khatib, 1993). Figure 14 summarizes the numerical analysis. In Figure 14a, we compare both methods in terms of the closest distances they approach a pedestrian. Closer distances negatively affect the comfort levels of pedestrians (Tipaldi & Arras, 2011). The comparison shows that our approach leads to less discomfort by leaving more space with the surrounding people. In Figure 14b, we present proxemics-zone intrusion counts for both methods and in Figure 14c, zone-intrusion durations are shown. These comparisons also show that our pipeline makes fewer and more brief intrusions of the proxemics zones of pedestrians. Lastly, in terms of distance-based metrics, these differences are statistically significant upon the application of the t-test with a confidence level of 0.95.

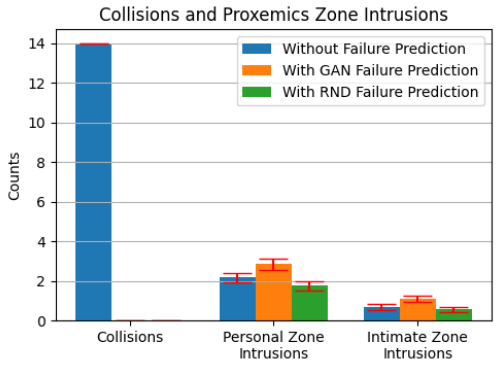


Figure 16

Performance of the Data-Driven Navigation System with and without the Failure Prediction Module in different environments with randomly placed pedestrians. Integration of the Failure Prediction Module successfully resolves collisions in each case. Refer to the text for more details.

Contribution of the Failure Prediction Module

The Failure Prediction Module that we incorporated into our navigation system aims to recognize potential extrapolating cases before any collision occurs. In the following, we first illustrate the contribution of this module. Later, we present an ablation study where we measure the performance of the complete pipeline with and without the Failure Prediction Module.

Figure 15a shows a failed extrapolation of the data-driven modules. The Failure Prediction Module detects such extrapolation cases and enables the navigation system to take measures, as shown in Figure 15b.

In order to demonstrate the contribution of the Failure Prediction Module to the Data-Driven Navigation System, we compare the performance of the entire navigation system with and without this module. Also, we present a comparison between the two outlier detection approaches on our system, namely the GAN and the RND. We ran 50 tests with randomly positioned pedestrians in the environment, just like the one shown in Figure 15. The numbers of collisions and proxemics-zone intrusions for three cases are shown in Figure 16.

Extrapolation errors occurred in 14 tests, and without the Failure Prediction Module such errors eventually led to collisions with pedestrians. On the other hand, the addition of the Failure Prediction Module enabled the successful prediction of all extrapolating cases with the exception on one case where RND notices the extrapolation too late for the robot to recover. Upon the prediction of extrapolation, the Hand-Crafted Reactive Controller Module temporarily took control of the mobile base, avoiding collisions altogether in all

instances. However, results show that the numbers and durations of the proxemics-zone intrusions are higher for the system with the Failure Prediction Module. There are two reasons for this situation. First, tests are finished early when a collision occurs; the robot could simply not complete tests with collisions. Therefore, it was not possible to examine the performance in the rest of the task leading to a low number of intrusions. Second, although the system with the Failure Prediction Module successfully executed social navigation in most cluttered cases, some of these environments were just too cluttered. In these cases, the navigation pipeline eluded collisions by turning them into zone intrusions.

Scalability of CNP

To evaluate the scalability of our controllers, we train the Data-Driven Local Controller Module with datasets of different sizes and perform simulated tests with the learned controllers. The results are given in Table 2. According to these tests, when the dataset size is small, when the size is $D = 50$ or $D = 250$, we see that the controller struggles to avoid collisions with pedestrians. The controller starts to learn meaningful avoidance behaviors when $D \geq 500$. As D increases, we see a decrease in the proxemics-zone intrusion counts. Average Displacement Error reaches meaningful levels only when $D > 1000$.

	Collision Count	**ADE**	**IZIC**	**PZIC**
D = 50	5	0.17 ± 0.02	5	5
D = 250	4	0.3 ± 0.03	5	6
D = 500	0	1.12 ± 0.02	5	10
D = 1000	0	0.25 ± 0.02	4	7
D = 1500	0	0.96 ± 0.02	0	0

Table 2

To evaluate how the size of the dataset affects the learning outcome of the Data-Driven Local Controller, we conduct a set of tests on simulation, with changing sizes (D) of expert demonstrations. Here, ADE refers to Average Displacement Error, while IZIC and PZIC refer to Intimate and Personal Zone Intrusion Counts. Please refer to the text for details.

Conclusion

This study presents a novel navigation system that can learn local and global navigation directly from expert demonstrations. We used a state-of-the-art deep learning framework, namely Conditional Neural Processes (CNPs), that can learn multimodal distributions around complex trajectories from relatively smaller datasets compared to its alternatives. One common problem with data-driven systems is that they are prone to extrapolation errors. Therefore, using only the data-driven architectures would eventually lead our system into collisions. To minimize this possibility, we devise a layered architecture inspired by the subsumption architecture (Brooks, 1986). We leverage state-of-the-art deep learning frameworks, namely Random Network Distillation (RND) and Generative Adversarial Networks (GANs), to work as arbitrators in this hierarchy. We have compared their performances and found that RND works better on average to detect out-of-distribution (OOD) samples enabling the system to infer the situations leading to extrapolation errors. When OOD states are encountered, the control of the robot is temporarily given to a manually encoded controller. We use an SFM-based controller for this purpose, but any socially-aware manually-encoded controller can be employed in the Hand-Crafted Reactive Controller.

In a simulated environment with a mobile robot, stationary and moving pedestrians, and given target points, we showed that our method could generate socially aware paths at the global level and successfully avoid pedestrians at the local level. This idea opposes the conventional approach of handling social navigation only in the local layer. An increase in the socialness of global trajectories improves the overall success of the social navigation frameworks. The results were verified in different simulated environments using metrics such as the average acceleration, path length, average displacement error, the closest distance to pedestrians and intrusion amount of their proxemics zones. These promising findings should be supported by real-world tests as reactions of the simulated people do not align perfectly with actual humans in natural environments. On the other hand, there are certain limitations of the current status of the system. An important limitation of this work is that the model does not consider the orientation and the velocity of the pedestrians. These factors would definitely alter the global and local plans of the robot. In future work, we plan to include these parameters in our models. Moreover, since we applied interpolation on the demonstration trajectories to get a continuous path, we lose the timing aspect of the trajectory in the Data Driven Global Controller Module. We are planning to solve this issue by adopting a different representation that enables us to learn global trajectories from the real-world data without interpolation so that we can produce more flexible global trajectories. There are recent benchmarking tools in the social robot navigation domain, such as Biswas, Wang, Silvera, Steinfeld, and Admoni (2022); Holtz and Biswas (2021); Tsoi, Hussein, Espinoza, Ruiz, and Vázquez (2020). They provide an automated evaluation of the algorithms and comparison with several conventional approaches, such as SFM and RVO (Van den Berg, Lin, & Manocha, 2008). We plan to use these tools to automate and expedite benchmarking efforts.

We believe that the social aspects of our approach can further benefit from the integration trajectory forecasting mechanisms, such as Gupta et al. (2018). In the future, we plan

to use future predictions about pedestrian trajectories as input to the trajectory calculation of the robot to create less disturbing trajectories by increasing the preference for uncongested parts of the environment, avoiding probable future encounters. Our system uses CNPs in both global and local controller modules. On the other hand, these two modules are independent of each other. We believe we can leverage the capability of CNPs in both layers, but some different approaches, like LSTMs or transformers, can also be used here. We also aim to investigate such techniques in the global controller layer in the future. Moreover, we aim to deploy this system on real robots. Since robots in the real-world need to be controlled in real-time for successful navigation, the challenge in applying our system is obtaining high-level parameters that we feed to CNPs in real-time. We believe that this can be addressed by leveraging the flexibility of CNPs to learn representations from any low-level and high-dimensional input, as shown in Gordon et al. (2019).

Acknowledgements

This work was supported by the BAGEP Award of the Science Academy. Thanks to Alper Ahmetoglu for helping with the implementation.

References

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (p. 961-971). doi: 10.1109/CVPR.2016.110

Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein gan.*

Asghari Oskoei, M., Walters, M., & Dautenhahn, K. (2010). An autonomous proxemic system for a mobile companion robot. In *Proceedings of the aisb 2010 symposium on new frontiers for human robot interaction.* Leicester, UK.

Biswas, A., Wang, A., Silvera, G., Steinfeld, A., & Admoni, H. (2022). Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction (THRI)*, *11*(3), 1–24.

Borenstein, J., Koren, Y., et al. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, *7*(3), 278–288.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, *2*(1), 14–23.

Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., ... Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artif. Intell.*, *114*, 3-55.

Cai, K., Wang, C., Cheng, J., De Silva, C. W., & Meng, M. Q.-H. (2020). Mobile Robot Path Planning in Dynamic Environments: A Survey. *arXiv preprint arXiv:2006.14195*.

Che, Y., Okamura, A. M., & Sadigh, D. (2020). Efficient and trustworthy social navigation via explicit and implicit robot–human communication. *IEEE Transactions on Robotics*, *36*(3), 692–707.

Chen, Y. F., Everett, M., Liu, M., & How, J. P. (2017). Socially aware motion planning with deep reinforcement learning. *CoRR*, *abs/1703.08862*. Retrieved from `http://arxiv.org/abs/1703.08862`

Dudek, G., & Jenkin, M. (2010). *Computational principles of mobile robotics*. Cambridge university press.

Farina, F., Fontanelli, D., Garulli, A., Giannitrapani, A., & Prattichizzo, D. (2017). Walking ahead: The headed social force model. *PloS one*, *12*(1), e0169734.

Ferrer, G., Garrell, A., & Sanfeliu, A. (2013). Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *2013 ieee/rsj international conference on intelligent robots and systems* (pp. 1688–1694).

Festo Robotics, R. (2020). *Robotino 4: For research and education.* Retrieved from `https://www.festo-didactic.com/int-en/learning-systems/factory-automation-industry-4.0/focus-trending-topics-i4.0/858/robotino-4-for-research-and-education.htm`

Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and autonomous systems*, *42*(3-4), 143–166.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, *4*(1), 23-33. doi: 10.1109/100.580977

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., ... Eslami, S. M. A. (2018, 10–15 Jul). Conditional Neural Processes. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 1704–1713). PMLR. Retrieved from `http://proceedings.mlr.press/v80/garnelo18a.html`

Giesbrecht, J. (2004). *Global path planning for unmanned ground vehicles* (Tech. Rep.). Defence Research and Development Suffield (Alberta).

Glasius, R., Komoda, A., & Gielen, S. C. (1995). Neural network dynamics for path planning and obstacle avoidance. *Neural Networks*, *8*(1), 125–133.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., & Turner, R. E. (2019). Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., & Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2255–2264).

Hall, E. (1966). *The hidden dimension*. New York, NY, US: Anchor Books.

Helbing, D., & Molnar, P. (1995). Social force model for pedestrian

dynamics. *Physical review E*, *51*(5), 4282.

Holtz, J., & Biswas, J. (2021). Socialgym: A framework for benchmarking social robot navigation. *arXiv preprint arXiv:2109.11011*.

Huang, K.-C., Li, J.-Y., & Fu, L.-C. (2010). Human-oriented navigation for service providing in home environment. In *Sice annual conference 2010, proceedings of* (pp. 1892–1897). Taipei, Taiwan.

Kambhampati, S., & Davis, L. (1986). Multiresolution path planning for mobile robots. *IEEE Journal on Robotics and Automation*, *2*(3), 135-145. doi: 10.1109/JRA.1986.1087051

Karnan, H., Nair, A., Xiao, X., Warnell, G., Pirk, S., Toshev, A., ... Stone, P. (2022). Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *arXiv preprint arXiv:2203.15041*.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 ieee international conference on robotics and automation* (Vol. 2, p. 500-505). doi: 10.1109/ROBOT.1985.1087247

Kim, B., & Pineau, J. (2016). Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, *8*(1), 51–66.

Kitani, K., Ziebart, B., Bagnell, J., & Hebert, M. (2012). Activity forecasting. *Computer Vision–ECCV 2012*, 201–214.

Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 ieee international conference on robotics and automation* (p. 1398-1404 vol.2). doi: 10.1109/ROBOT.1991.131810

Kretzschmar, H., Spies, M., Sprunk, C., & Burgard, W. (2016). Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, *35*(11), 1289–1307.

Kruse, T., Pandey, A. K., Alami, R., & Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, *61*(12), 1726–1743.

Kuderer, M., Kretzschmar, H., Sprunk, C., & Burgard, W. (2012). Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems.*

Lam, C.-P., Chou, C.-T., Chang, C.-F., & Fu, L.-C. (2010). Human-centered robot navigation—toward a harmoniously coexisting multi-human and multi-robot environment. In *Intelligent robots and systems (iros), 2010 ieee/rsj international conference on* (pp. 1813–1818). Taipei, Taiwan.

Latombe, J. (1991). *Robot motion planning: Edition en anglais*. Springer. Retrieved from `https://books.google.com.tr/books?id=Mbo_p4-46-cC`

Lerner, A., Chrysanthou, Y., & Lischinski, D. (2007). Crowds by example. In *Computer graphics forum* (Vol. 26, pp. 655–664).

Levine, S., Popovic, Z., & Koltun, V. (2011). Nonlinear inverse reinforcement learning with gaussian processes. *Advances in neural information processing systems*, *24*, 19–27.

Manso, L. J., Nuñez, P., Calderita, L. V., Faria, D. R., & Bachiller, P. (2020). Socnav1: A dataset to benchmark and learn social navigation conventions. *Data*, *5*(1), 7.

Martin-Martin, R., Patel, M., Rezatofighi, H., Shenoi, A., Gwak, J., Frankel, E., ... Savarese, S. (2021). Jrdb: A dataset and benchmark of egocentric robot visual perception of humans in built environments. *IEEE transactions on pattern analysis and machine intelligence*.

Mavrogiannis, C., Baldini, F., Wang, A., Zhao, D., Trautman, P., Steinfeld, A., & Oh, J. (2021). Core challenges of social robot navigation: A survey. *arXiv preprint arXiv:2103.05668*.

Mead, R., Atrash, A., & Matarić, M. J. (2011). Proxemic feature recognition for interactive robots: Automating metrics from the social sciences. In *Social robotics* (pp. 52–61). Springer.

Murphy, R. R. (2019). *Introduction to ai robotics*. MIT press.

Nonaka, S., Inoue, K., Arai, T., & Mae, Y. (2004). Evaluation of human sense of security for coexisting robots using virtual reality. 1st report: evaluation of pick and place motion of humanoid robots. In *Ieee international conference on robotics and automation, 2004. proceedings. icra'04. 2004* (Vol. 3, pp. 2770–2775).

Nourbakhsh, I., Kunz, C., & Willeke, T. (2003). 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)* (Vol. 4, p. 3636-3641 vol.3). doi: 10.1109/IROS.2003.1249720

Okal, B., & Arras, K. O. (2016). Formalizing normative robot behavior. In *International conference on social robotics* (pp. 62–71).

Orebäck, A., & Christensen, H. I. (2003). Evaluation of architectures for mobile robotics. *Autonomous robots*, *14*(1), 33–49.

Pellegrini, S., Ess, A., Schindler, K., & Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 ieee 12th international conference on computer vision* (pp. 261–268).

Pérez-Higueras, N., Caballero, F., & Merino, L. (2018). Learning human-aware path planning with fully convolutional networks. In *2018 ieee international conference on robotics and automation (icra)* (pp. 5897–5902).

Quinlan, S., & Khatib, O. (1993). Elastic bands: Connecting path planning and control. In *[1993] proceedings ieee international conference on robotics and automation* (pp. 802–807).

Rohmer, E., Singh, S. P. N., & Freese, M. (2013). CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework. In *Proc. of the international conference on intelligent robots and systems (iros).* (www.coppeliarobotics.com)

Rosmann, C., Hoffmann, F., & Bertram, T. (2015). Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control. In *2015 european control conference (ecc)* (p. 3352-3357). doi: 10.1109/ECC.2015.7331052

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1–48.

Syrdal, D. S., Koay, K. L., Walters, M. L., & Dautenhahn, K. (2007). A personalized robot companion-the role of individual differences on spatial preferences in hri scenarios. In *Robot and human interactive communication, 2007. ro-man 2007. the 16th ieee international symposium on* (pp. 1143–1148). Jeju Island, Korea.

Tai, L., Zhang, J., Liu, M., & Burgard, W. (2018). Socially compli-

ant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 ieee international conference on robotics and automation (icra)* (pp. 1111–1117).

Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., ... others (2000). Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, *19*(11), 972–999.

Tipaldi, G. D., & Arras, K. O. (2011). Please do not disturb! minimum interference coverage for social robots. In *2011 ieee/rsj international conference on intelligent robots and systems* (pp. 1968–1973).

Trautman, P., & Krause, A. (2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 ieee/rsj international conference on intelligent robots and systems* (pp. 797–803).

Tsoi, N., Hussein, M., Espinoza, J., Ruiz, X., & Vázquez, M. (2020). Sean: Social environment for autonomous navigation. In *Proceedings of the 8th international conference on human-agent interaction* (pp. 281–283).

Vadakkepat, P., Tan, K. C., & Ming-Liang, W. (2000). Evolutionary artificial potential fields and their application in real time robot path planning. In *Proceedings of the 2000 congress on evolutionary computation. cec00 (cat. no. 00th8512)* (Vol. 1, pp. 256–263).

Van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 ieee international conference on robotics and automation* (pp. 1928–1935).

Vasquez, D., Okal, B., & Arras, K. O. (2014). Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In *2014 ieee/rsj international conference on intelligent robots and systems* (pp. 1341–1346).

Vemula, A., Muelling, K., & Oh, J. (2018). Social attention: Modeling attention in human crowds. In *2018 ieee international conference on robotics and automation (icra)* (pp. 4601–4607).

Wulfmeier, M., Ondruska, P., & Posner, I. (2015). Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*.

Yan, Z., Duckett, T., & Bellotto, N. (2017, September). Online learning for human classification in 3d lidar-based tracking. In *In proceedings of the 2017 ieee/rsj international conference on intelligent robots and systems (iros)*. Vancouver, Canada.

Zanlungo, F., Ikeda, T., & Kanda, T. (2011). Social force model with explicit collision prediction. *EPL (Europhysics Letters)*, *93*(6), 68005.

Zhu, Q., Yan, Y., & Xing, Z. (2006). Robot path planning based on artificial potential field approach with simulated annealing. In *Sixth international conference on intelligent systems design and applications* (Vol. 2, pp. 622–627).

Appendix A
Datasets

Real-World Data

We trained the model on a real-world dataset, namely SCAND (Karnan et al., 2022). This dataset contains more than 8 hours of navigation trajectories in social environments, recorded using a teleoperated robot. Along these trajectories, researchers also present the raw data from several sensors, including cameras, lidars, GPS, etc.

In this dataset, first, we have manually extracted the navigation data of 30 social scenarios where the robot encounters a pedestrian and avoids collision by maneuvering. In each encounter, we manually annotated the positions of the pedestrians at multiple time steps. At the same time, we obtained the robot's positions using the GPS sensor and recorded this information with corresponding time stamps. Later, we applied interpolation to position data to get the robot's and pedestrians' continuous trajectories. As in Pérez-Higueras, Caballero, and Merino (2018), where researchers use robot-centric data to learn social navigation, we processed the position data of timed trajectories to convert it to egocentric. Then, we applied rotational transformations to egocentric trajectories for data augmentation purposes. Data augmentation is the process of increasing the size of the dataset artificially. This technique has been used in many deep learning applications widely (Shorten & Khoshgoftaar, 2019). The entire data-processing procedure is depicted in Figure A1.

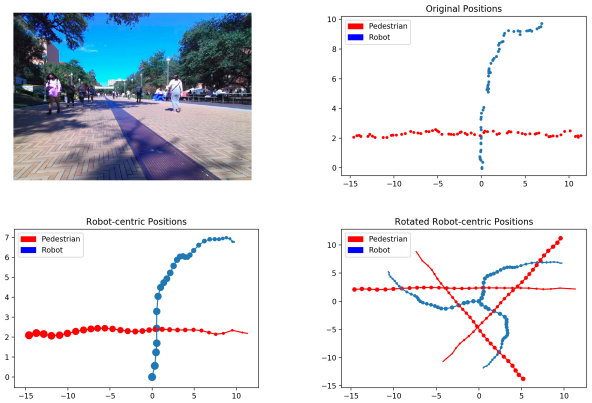


Figure A1

Processing the data from SCAND (Karnan et al., 2022). First, raw sensor data of the selected trajectory clips with evasive maneuvers are gathered. Robot and pedestrian positions are annotated on the data and transformed into a robot-centric coordinate frame. Then, interpolation is applied to obtain entire trajectories. Finally, rotation is applied to increase the size of the dataset.

Simulated Data

To gather a set of socially-compliant demonstration trajectories for learning, we implemented the Social Force Model, described in Helbing and Molnar (1995) to control the robot. Assuming that it generates socially plausible trajectories, we recorded 1500 trajectories with random start, goal, and pedestrian positions. In each trial, single and multiple stationary and dynamic pedestrians are placed at random positions. Figure A2 shows snapshots of this procedure from the simulator.

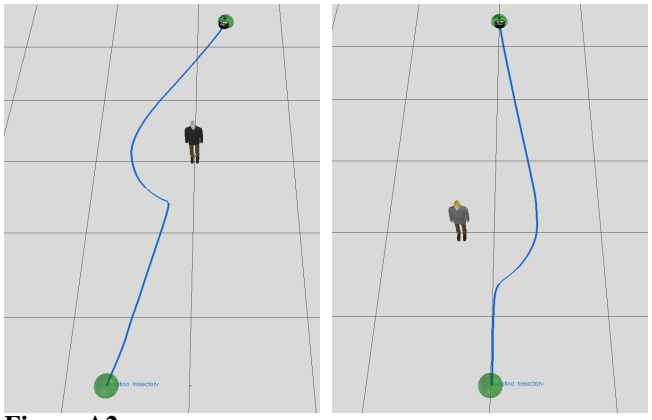


Figure A2

Data collection on the simulation. The robot implements Social Force Model to avoid randomly placed pedestrians. Green spheres show the navigation task's start and goal positions, and the blue line shows the motion trajectory.

Appendix B
Models

The implementation of the model and the training data can be found at `https://github.com/yildirimyigit/cnmp`. In addition, data-processing scripts, ROS nodes, and environments used in the simulator can be found at `https://github.com/yildirimyigit/irl_sfm`.

In the following, we present the control parameters of all models used throughout the study.

Social Force Model

Hyperparameter	Value
Relaxation Time	2.3
Force Strength	6.40
Force Range	0.25
Maximum Velocity	2.5

Table B1

Model parameters of SFM

Data-Driven Global Controller (Conditional Neural Process)

Hyperparameter	Value
Maximum Number of Observations	10
Number of Hidden Layers	3
Width of Layers of Encoder Network	256, 256, 256
Width of Layers of Query Network	256, 256, 256
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4

Table B2

Hyperparameters of the CNP of the Data-Driven Global Controller

Data-Driven Local Controller (Conditional Neural Process)

Hyperparameter	Value
Maximum Number of Observations	20
Number of Hidden Layers	3
Width of Layers of Encoder Network	256, 384, 512
Width of Layers of Query Network	512, 384, 256
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4

Table B3

Hyperparameters of the CNP of the Data-Driven Local Controller

Feed-Forward Neural Network

Feed-Forward Neural Network is used for performance comparison with CNP.

Hyperparameter	Value
Number of Hidden Layers	3
Width of Hidden Layers	256, 512, 1024
Optimizer	SGD
Activation Function	ReLU
Learning Rate	1e-3

Table B4

Hyperparameters of the Feed-Forward Neural Network as the global controller

Failure Prediction Module - GAN

Hyperparameter	Value
Number of Hidden Layers	2
Width of Hidden Layers	128, 128
Generator Noise	64 Dimensional, Diagonal Gaussian
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4

Table B5

Hyperparameters used in the GAN of Failure Prediction Module

Failure Prediction Module - RND

Hyperparameter	Value in Target	Value in Predictor
Number of Hidden Layers	3	3
Width of Hidden Layers	512,512,512	512,512,512
Optimizer	N/A	Adam
Activation Function	ELU	ELU
Learning Rate	N/A	1e-4

Table B6

Hyperparameters used in the RND of Failure Prediction Module