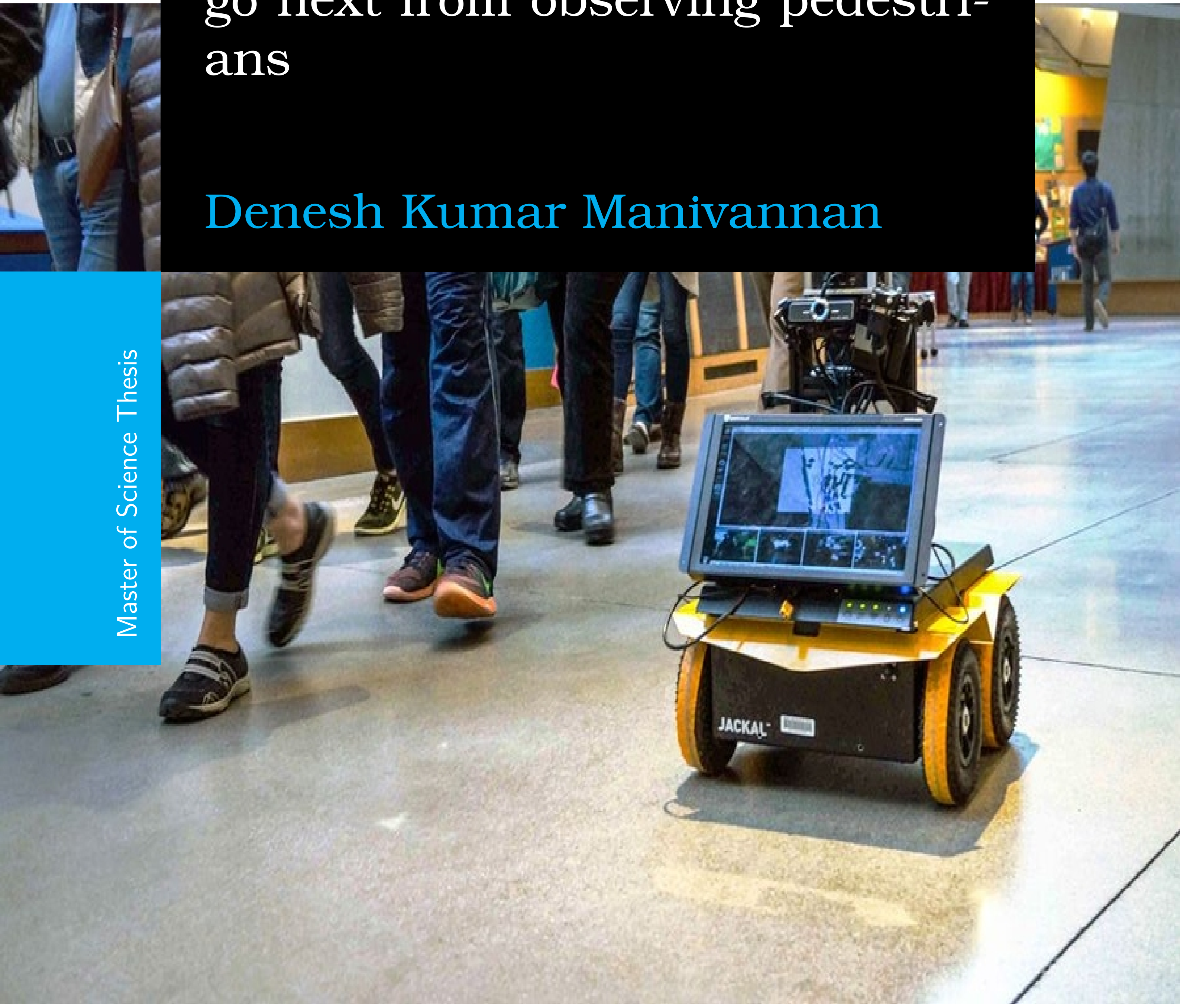
Continuously learning where to go next from observing pedestrians

Denesh Kumar Manivannan

Master of Science Thesis



TU Delft Delft University of Technology

Cognitive Robotics

Continuously learning where to go next from observing pedestrians

Master of Science Thesis

For the degree of Master of Science in Robotics at Delft University of Technology

Thesis Committee :

Dr. Javier Alonso Mora
Dr. Laura Ferranti
Dr. Luca Laurenti
Msc. Luzia Knoedler

Institution : Delft University of Technology

Place: Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft

Program: MSc. Robotics

November 22, 2022

Cover Image credits: MIT Jackal robot

Delft University of Technology
Department of
Cognitive Robotics (CoR)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Continuously learning where to go next from observing pedestrians

by

Denesh Kumar Manivannan

in partial fulfillment of the requirements for the degree of

Master of Science Robotics

Dated: November 22, 2022

Supervisor(s): Dr. Javier Alonso Mora

Msc. Luzia Knoedler

Reader(s): Dr. Laura Ferranti

Dr. Luca Laurenti

Abstract

Socially compliant robot navigation in pedestrian environments remains challenging owing to uncertainty in human behavior and varying pedestrian preferences in different social contexts. Local optimization planners like Model Predictive Control can incorporate collision avoidance constraints, but they can only lead to socially compliant trajectories if the cost function embeds information about the desired social behavior. The same holds for Reinforcement Learning, where a sophisticated reward function needs to be defined. However, formalizing social behavior through a reward or cost function is difficult due to the complex nature of pedestrian behavior. Imitation learning allows for inferring the desired behavior by learning from human demonstrations, making them suitable for learning socially compliant navigation policies but without any safety considerations. In this work, we propose to learn a socially compliant navigation policy directly by observing surrounding pedestrians' trajectories from a commonly available detection and tracking pipeline and combine it with a local optimization planner to enhance safety. A Subgoal Recommender policy is developed to guide the local optimization planner to generate socially compliant trajectories by providing intermediate subgoals. To adapt the policy to changing social contexts without forgetting previously learned information, we train the Subgoal Recommender in a Continual Learning setup exploiting new pedestrian data.

We demonstrate in simulation that our method can learn a policy that has similar performance metrics as that of the observed trajectories with 95% confidence estimated from a t-test, resulting in a lesser number of collisions. Further, the policy can adapt to different social preferences exhibited by pedestrians, while being able to remember the learned behavior in a previously encountered social context. Furthermore, we show that our proposed method can learn navigation policies from actual pedestrian data recorded using the onboard perception pipeline of a Clearpath Jackal robot.

Acknowledgements

I want to thank Dr. Javier Alonso Mora for granting me the opportunity to work in the Autonomous Mobile Robots Lab and supervising my thesis research. His feedback and support helped me through all stages of the thesis. I am incredibly grateful to my daily supervisor Luzia. She has always been there for me, answering all my queries and having long, interesting discussions that were valuable for completing my work. I want to thank my parents, Manivannan R, Vimala D, and my sister Lakshmi M who have been a constant beacon of support throughout my life. Without them, my journey in TU Delft would not have been possible. With their love and support, I have been able to achieve my dreams.

Finally, I would like to thank my friends Anish, Barry, Dhruv, Dinesh, Francesco, Iva, Katherina, Ravi, Rajneesh, Renga, Steven, Tarun, Revanth, Varshiny and Vivek, who have provided me with a lot of the support I needed throughout my TU Delft life, and made the last couple of years more fun than I could imagine.

Chapter 1

Scientific paper

Continuously learning where to go next from observing pedestrians

1st Denesh Kumar Manivannan
Cognitive Robotics Dept.
TU Delft
Delft, Netherlands

***Abstract*—Socially compliant robot navigation in pedestrian environments remains challenging owing to uncertainty in human behavior and varying pedestrian preferences in different social contexts. Local optimization planners like Model Predictive Control can incorporate collision avoidance constraints, but they can only lead to socially compliant trajectories if the cost function embeds information about the desired social behavior. The same holds for Reinforcement Learning, where a sophisticated reward function needs to be defined. However, formalizing social behavior through a reward or cost function is difficult due to the complex nature of pedestrian behavior. Imitation learning allows for inferring the desired behavior by learning from human demonstrations, making them suitable for learning socially compliant navigation policies but without any safety considerations. In this work, we propose to learn a socially compliant navigation policy directly by observing surrounding pedestrians' trajectories from a commonly available detection and tracking pipeline and combine it with a local optimization planner to enhance safety. A Subgoal Recommender policy is developed to guide the local optimization planner to generate socially compliant trajectories by providing intermediate subgoals. To adapt the policy to changing social contexts without forgetting previously learned information, we train the Subgoal Recommender in a Continual Learning setup exploiting new pedestrian data.**

We demonstrate in simulation that our method can learn a policy that has similar performance metrics as that of the observed trajectories with 95% confidence estimated from a t-test, resulting in a lesser number of collisions. Further, the policy can adapt to different social preferences exhibited by pedestrians, while being able to remember the learned behavior in a previously encountered social context. Furthermore, we show that our proposed method can learn navigation policies from actual pedestrian data recorded using the onboard perception pipeline of a Clearpath Jackal robot.

***Index Terms*—Social Compliance, Subgoal Recommender, Robot navigation, Imitation Learning**

I. Introduction

The applications of mobile robots in human-centered environments are becoming increasingly popular, ranging from telepresence robots in offices [1] to robots that transport sterile goods in hospitals [2], [3]. Workplace environments benefit from such robots as they can transport equipment efficiently between different places, thus obviating the need for a dedicated person to carry heavy objects constantly. Integrating robots within workplace environments is challenging since the robots should navigate seamlessly without distracting or affecting the employees' daily routines. For robots to be accepted, they must not only avoid collisions but also behave in a socially compliant way by maintaining a comfortable distance, following social norms like passing on the right, and showing similar low-level behaviors to humans [4].

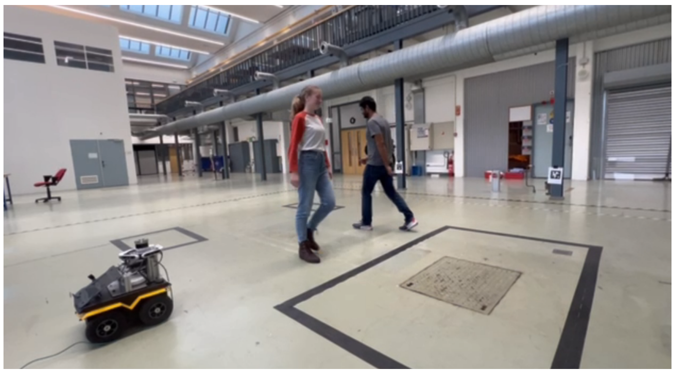


Fig. 1: Jackal robot used for recording pedestrian trajectory data at 3me faculty, TU Delft. The available detection and tracking pipeline can be used to observe the surrounding pedestrians continuously and adapt the behavior to new social contexts.

Socially compliant navigation for mobile robots is an active area of research. Previous works incorporating social compliance in navigation algorithms include methods that use Model-based techniques [5]–[8], Joint Planning [9]–[14], Deep Reinforcement Learning (DRL) [15]–[19] and Imitation Learning (IL) [20]–[29]. Model-based methods are interpretable and computationally efficient since they are mostly based on physics or geometry and can be expressed with simple mathematical equations. However, they do not paint an accurate picture of reality since they make many assumptions in modeling human navigation behavior. Joint Planning methods are similar to model-based methods as they assume some mathematical model for human navigation. However, they are computationally expensive since they jointly plan trajectories for all the agents in the scene. On the other hand, DRL algorithms assume that humans aim to optimize a certain reward function and thus learn a policy only by maximizing the rewards. However, since it is difficult to embed complex human behavior in hand-crafted rewards, the learned navigation policies do not necessarily display the desired level of social compliance. Furthermore, RL agents are usually trained in simulations, where the learned behavior is highly dependant on the simulated pedestrians' models. In contrast, RL algorithms trained with real humans cannot be used owing to safety concerns. IL algorithms alleviate these issues by learning policies directly from expert demonstrations. Nevertheless, IL algorithms are limited by the amount of training data since expert demonstrations are expensive. Thus, we propose to observe the trajectories of the surrounding pedestrians instead of having an expert drive the robot to obtain sufficient demonstrations to train IL algorithms.

Current literature on socially compliant navigation algorithms cannot guarantee to satisfy important environmental and collision avoidance constraints with other dynamic agents such as humans in the scene. The difference in robot and human dynamics is also not considered while learning navigation policies from human demonstrations. Model Predictive Control (MPC) has been long used in industrial mobile robots for trajectory tracking [30] since such constraints can be specified explicitly to solve an optimization problem that calculates feasible input commands. The performance of MPC algorithms is limited by their planning horizon and cost function in the optimization problem formulation. Hence, many recent works started combining learning-based methods with MPC to enhance closed-loop performance by using learning algorithms such as RL and use MPC for constraint satisfaction [31]–[35]. We propose to learn a socially compliant guidance policy from human trajectory data to enhance MPC by considering social interactions beyond the prediction horizon and thus improve performance and safety.

Another limitation of existing methods is that the policies are learned offline and then deployed on the robot. Hence, such static policies cannot adapt to environments that demand varying levels of social compliance. For instance, in certain places (most European countries and the USA), people prefer to keep on the right side while crossing, while in other places (UK and India), people prefer the left side, which makes it essential for the robot to adapt its navigation policy to match the environment.

The main contributions of this work include:

- A method to learn a socially compliant guidance policy that recommends intermediate subgoals to the robot by observing pedestrian trajectories, which can be combined with MPC to satisfy the robot's kinodynamic and collision avoidance constraints.
- A Continual Learning framework to adapt the learned socially compliant navigation policy to different social contexts by observing surrounding pedestrians in new environments without forgetting previously obtained knowledge.

A. Related work

1) Socially compliant navigation: Kruse et al. [4] identify Comfort, Naturalness and Sociability as the three principal components of socially compliant navigation. Previous works on socially compliant navigation incorporate one or more of these components in the algorithms. Model-based algorithms use hand-crafted fixed models inspired by physics [9], [13], [14], [36], or geometry [37]–[39] to incorporate comfort and sociability in the policy and enforce social compliance. On the other hand, joint planning-based methods [10]–[12] solve a joint optimization problem where the planner assumes a model for the rest of the agents and plans the robot's trajectory by introducing constraints (minimum distance between agents, smoothness) to induce socially compliant behavior. Many approaches for socially compliant navigation using DRL are based on Collision Avoidance with Deep Reinforcement Learning (CADRL) [40] where the cooperation/interaction behavior between the agents is embedded within the value function. The following works suggest modifications to the reward function to induce navigation behavior closer to that of humans by encoding social norms [16], architectural changes to consider human-robot interactions [17] and human-human interactions [15]. IL for socially compliant navigation has gained much traction recently since the policy incorporates all components of social compliance by directly learning from expert demonstrations. Previous works use Behavior Cloning (BC) [41]–[44], where the policy is learned in a supervised learning setting, Inverse Reinforcement Learning [24], [45]–[47], in which the reward function that the humans are thought to maximize is learned to train the policy in an RL framework, and finally Adversarial learning methods [21], [27], [48], [49], where a generator network competes against a discriminator to learn policies that result in trajectories very similar to that of the human experts. However, most of these works directly predict input commands to be followed by the robot without consideration for different robot dynamics and do not focus on safety considerations. Further, the performance of the policies in different social scenarios needs to be evaluated in environments with varying degrees of social compliance.

2) Learning based methods to guide MPC: Optimization-based planners like MPC are attractive in safety-critical applications because they can incorporate safety constraints into their framework. However, the performance of classical MPC algorithms is limited by the planning horizon. A low planning horizon leads to sub-optimal short-sighted decisions, while longer planning horizons lead to high computational costs. Hence, many works propose to use learning-based policies to enhance the performance of MPC. For instance, the typical quadratic cost-to-go function is replaced with value functions learned by RL in [34], [35]. Instead, we propose to have a higher-level policy that computes intermediate subgoals followed by the classical MPC controller that calculates input commands at each instant. The concept of using subgoals to guide local planners was introduced in [33], where a policy is trained in an RL framework to recommend subgoals to the MPC. However, no special emphasis on social compliance was given in the reward function as it only focused on discouraging collisions and reaching the goal efficiently. Since human navigation behavior is difficult to embed in hand-crafted reward functions, we propose to use Imitation learning (IL) to learn a policy to recommend subgoals by taking social interactions beyond MPC's prediction horizon into account, thus comprising the components of social compliance described in I-A1 as well as safety considerations from the MPC.

3) Continual Learning: Traditional Deep Learning (DL) methods focus on training models by assuming a fixed data distribution. The performance of these static models is bound to deteriorate with a shift in the data distribution of test samples after deployment [50], [51]. An easy solution to handle non-stationary data can be to collect new training samples and retrain the network by aggregating them with previous data. However, over the longer run, data aggregation could be more computationally intense as the number of training samples would become too high. Another possibility is to initialize the network parameters obtained by training on the previous tasks and retrain using the latest training samples alone. However, the previously learned information is affected while adapting the network parameters to learn new tasks. This phenomenon is termed catastrophic forgetting [52]. The branch of DL algorithms that proposes to adapt models to dynamic data distributions without aggregating all previous data while retaining previous knowledge is called Continual Learning (CL) [53]. There are three main approaches in CL to prevent catastrophic forgetting: Rehearsal [54]–[56], where few data samples representative of the older data distributions are stored and replayed; Regularization [52], [57], [58], that involves adding a regularization term to the loss function to restrict changes to parameters that are instrumental for previous tasks performance and Architectural modifications [59], [60], where new layers are added to the existing architecture to improve performance in new tasks while preserving the layers that are responsible for the performance in previous tasks. Earlier works on CL assume the availability of information about when the shift in data distribution occurs. Recent works on Online Continual Learning (OCL) relax the assumption of knowing the task label beforehand and propose methods to infer the distributional change by calculating network parameters' sensitivity to learning new information [61], storing intermediate latent layers for replaying [62] and using Gaussian mixture parameters in the architecture [63]. The tasks mentioned above correspond to different social scenarios, and we assume that the information about when the policy needs to adapt is available to the robot. We propose a framework to train a socially compliant navigation policy in a CL setting, where the policy is adapted by observing the surrounding pedestrians in the scene, which is yet to be attempted to the best of our knowledge.

II. Preliminaries

Vectors are denoted in bold lower case letters ($\mathbf{x}$) and matrices are capitalized (A), $||\mathbf{y}||$ represents the Euclidean norm of vector $\mathbf{y}$.

A. Socially compliant motion planning problem formulation

The robot's goal is to navigate from an initial position $\mathbf{p}^0$ to a goal position $\mathbf{g}$ in the $\mathcal{R}^2$ plane. Let us consider 'n_t' people in the robot's field of view at a particular instant 't.' $\mathbf{s}_t$ denotes the robot's state, while $\mathbf{s}_t^i$ denotes the state of the i^{th} closest pedestrian at time t. $\mathbf{z}_t = [\mathbf{s}_t^1\ \mathbf{s}_t^2 \ldots \mathbf{s}_t^{n_t}]$ comprises the states of all n_t humans. Though the position and velocity of humans are easy to observe, they have internal states like the intended goal position and preferred speed, which are not observable by the robot. Hence, it is essential to note that $\mathbf{z_t}$ corresponds to the observable state information of other humans in the scene. The joint state for the robot navigation problem is given by $\mathbf{s}_t^{jn} = [\mathbf{s}_t,\ \mathbf{z}_t]$. Let the robot's action be denoted by $\mathbf{u}_t$ (for example, velocity). The objective to find a policy that leads to trajectories that are similar to humans while ensuring collision-free motion is formally defined as:

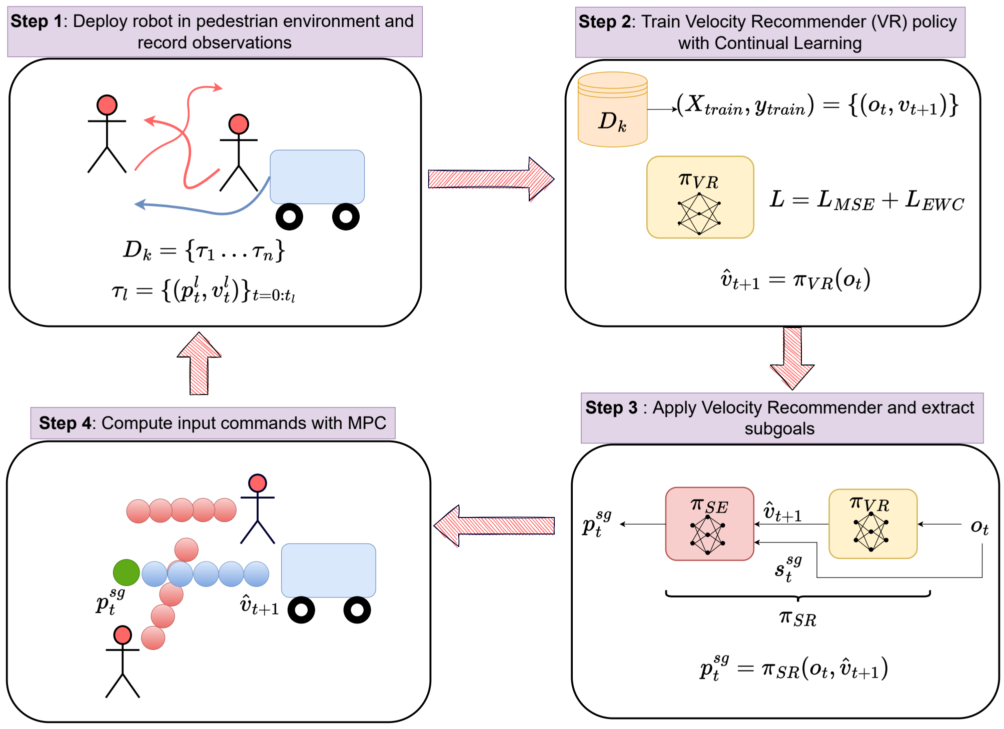
$$\begin{aligned}
\pi^* = \underset{\pi}{\operatorname{argmin}} \sum_{t=1}^{T} ||f(\mathbf{p}_{t-1}, \pi(\mathbf{s}_{t-1}^{jn})) - \mathbf{h}_t^{pos}||_2 \quad & (1a)\\
\text{s.t. } \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{u}_t) \quad & (1b)\\
\mathbf{h}_{t+1} = f_h(\mathbf{h}_t) \quad & (1c)\\
\mathbf{u}_t = \pi(\mathbf{s}_{t-1}^{jn}) \quad & (1d)\\
\mathbf{p}_0 = \mathbf{h}_0^{pos} = \mathbf{p}^0 \quad & (1e)\\
\mathbf{p}_T = \mathbf{h}_T^{pos} = \mathbf{g} \quad & (1f)\\
\mathcal{O}(\mathbf{s}_t) \cap \mathcal{O}_t^i = \emptyset \quad & (1g)\\
\mathbf{u}_t \in \mathcal{U}, \mathbf{s}_t \in \mathcal{S} \quad & (1h)\\
\forall t \in [0, T], \quad \forall i \in \{1, \ldots, n_t\} &
\end{aligned}$$

where $\mathbf{p}_t$ and $\mathbf{h}_t^{pos}$ denote positions of the robot and the average human agent at time t. T is the episode length or in other words, the time limit imposed in the optimization problem to reach the goal. The objective function (1a) consists of the accumulation of Euclidean distance between the position reached by the robot following policy $\pi(.)$ and the position reached by the average human agent. Equation (1b) refers to the transition dynamics constraints of the robot considering the dynamic model f, (1c) refers to the transition dynamics constraints of a human considering the dynamic model f_h. The policy π maps the joint state configuration of the system to the robot's action as defined in (1d). $\mathcal{O}$ represents the area occupied by the robot while $\mathcal{O}_t^i$ is the area occupied by the i^{th} agent at time t. The initial state constraints, terminal constraints, collision avoidance and admissible states and inputs constraints are given by (1e), (1f), (1g) and (1h) respectively. We assume that all the other agents follow the same policy like in [40].

B. Agent Dynamics

We assume that the human agents follow a first-order Linear Dynamics model [64] as the human body is very agile, and the delay in executing the desired velocity commands is negligible. However, the real robot has a considerable time delay due to lesser degrees of freedom. Thus, we assume a second-order unicycle model [65] for the robot that takes into account the time delay by considering the linear and angular acceleration as the input given by:

$$\begin{aligned}
&\dot{x}_t = V_t \cos\theta_t \quad \dot{V}_t = u_a\\
&\dot{y}_t = V_t \sin\theta_t \quad \dot{\omega} = u_\alpha \qquad (2)\\
&\dot{\theta}_t = \omega
\end{aligned}$$



Step 1: Deploy robot in pedestrian environment and record observations
$D_k = \{\tau_1 \dots \tau_n\}$
$\tau_l = \{(p_t^l, v_t^l)\}_{t=0:t_l}$
Step 2: Train Velocity Recommender (VR) policy with Continual Learning
D_k
$(X_{train}, y_{train}) = \{(o_t, v_{t+1})\}$
π_{VR}
$L = L_{MSE} + L_{EWC}$
$\hat{v}_{t+1} = \pi_{VR}(o_t)$
Step 3 : Apply Velocity Recommender and extract subgoals
p_t^{sg}
π_{SE}
$\hat{v}_{t+1}$
π_{VR}
o_t
s_t^{sg}
π_{SR}
$p_t^{sg} = \pi_{SR}(o_t, \hat{v}_{t+1})$
Step 4: Compute input commands with MPC
p_t^{sg}
$\hat{v}_{t+1}$

Fig. 2: The proposed framework to continually recommend subgoals.

where x_t and y_t are the x and y coordinates of robot's position, θ_t is the heading angle in a global frame. The robot's forward velocity is given by V_t , while ω denotes the angular velocity and $\mathbf{u_t} = [u_a \;\; u_\alpha]$ is the input containing linear and angular acceleration respectively.

C. Model Predictive Control formulation

Brito et al. [33] propose to use DRL to learn intermediate subgoals to provide global guidance to an MPC motion planner. In this subsection, We present the MPC formulation to calculate appropriate robot input commands to follow the subgoal denoted by $\mathbf{p}_t^{sg}$. The observation $\mathbf{o}_t$ contains the robot's state $\mathbf{s}_t$ and the combined states z_t of the n_t surrounding humans. The action of the guidance policy is defined as a position increment δ_t providing the direction maximizing the robot's rewards. Based on δ_t, the current subgoal to the robot is:

$$\mathbf{p}_t^{sg} = \mathbf{p}_t + \delta_t \tag{3}$$

The guidance policy is trained by considering a reward function with a positive reward r_{goal} for being at the goal $\mathbf{g}$, a negative reward $r_{\text{collision}}$ for being in a collision and a negative reward r_t otherwise. The subgoal $\mathbf{p}_t^{sg}$ is utilized in the MPC cost function, which ensures that the calculated input commands takes the robot's position to the subgoal within the prediction horizon with minimal efforts while dynamic feasibility and collision avoidance constraints are satisfied. The cost is composed of a terminal cost J_N after N steps with the weight coefficient Q_N, given by:

$$J_N(\mathbf{p}_0, \mathbf{p}_N, \mathbf{p}_0^{sg}) = \left\| \frac{\mathbf{p}_N - \mathbf{p}_0^{sg}}{\mathbf{p}_0 - \mathbf{p}_0^{sg}} \right\|_{Q_N} \tag{4}$$

and stage costs $J_t^u(u_t) = \|\mathbf{u}_t\|_{Q_u}$ with weight coefficient Q_u resulting in the following non-convex optimization problem:

$$\begin{aligned} \min_{\mathbf{s}_{1:N}, \mathbf{u}_{0:N-1}} \quad & J_N(\mathbf{p}_0, \mathbf{p}_N, \mathbf{p}_0^{sg}) + \sum_{t=0}^{N-1} J_t^u(\mathbf{u}_t) \\ \text{s.t.} \quad & \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{u}_t), \\ & \left\|\mathbf{p}_t - \mathbf{p}_t^i\right\| > r + r^i, \\ & \mathbf{u}_t \in \mathcal{U}, \quad \mathbf{s}_t \in \mathcal{S}, \\ & \forall i \in \{1, \dots, n\}; \forall t \in \{0, \dots, N-1\}. \end{aligned} \tag{5}$$

where n is the maximum number of other agents considered for MPC's calculation since it is computationally infeasible to include all agents in the scene for the optimization problem.

III. Method

We present a four-level architecture for socially compliant navigation consisting of an Observations collection, a Velocity Recommender, a Subgoal Extractor and a local motion planner (MPC). The overview of our framework can be seen in Fig 2. The first step shown involves the robot being deployed in a pedestrian environment where it can either be stationary or follow some initial policy to collect data by observing pedestrians' trajectories. In the second step, the sensor observations and corresponding pedestrian velocities are used

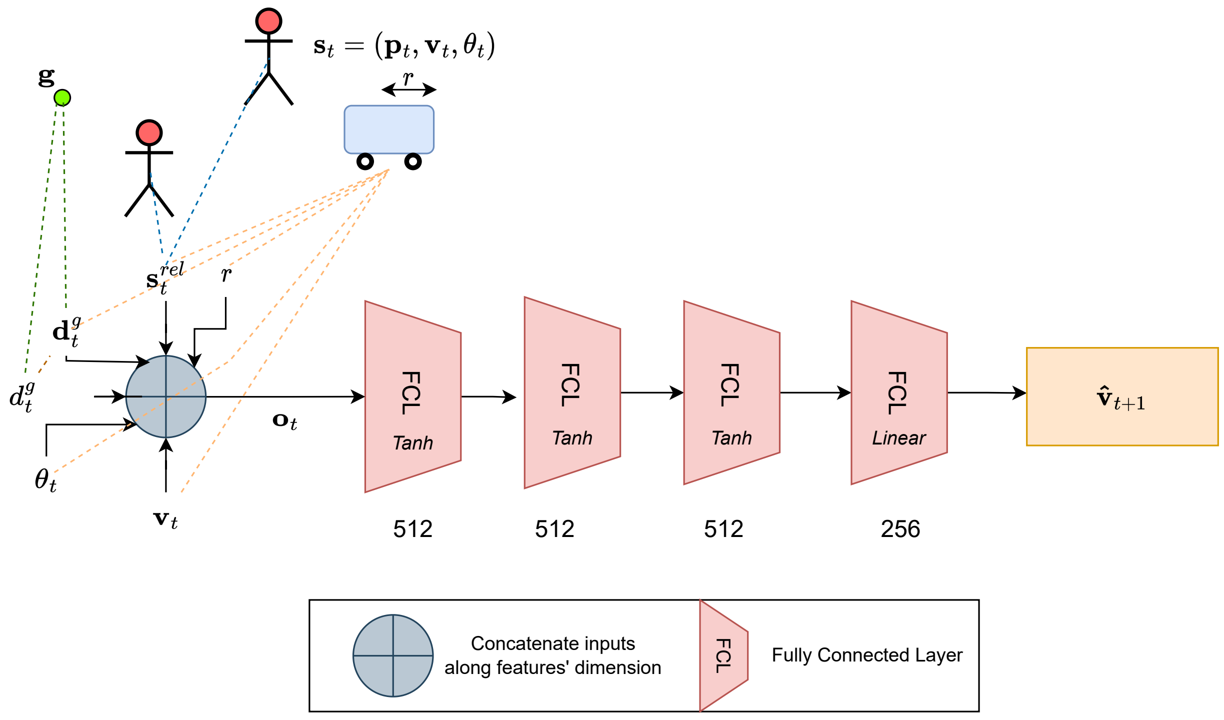


Fig. 3: Velocity Recommender architecture diagram

to train the "Velocity Recommender" (VR) network. We explain the process of training the VR policy, taking in sensor observations as input and predicting the velocity that a human would have preferred for that observation in Subsec III-A. However, since the robot may have different dynamics from a human, the predicted velocity doesn't necessarily lead to the same position. Hence, the next step is to extract a subgoal from the recommended velocity using the "Subgoal Extractor" (SE) network that accounts for the robot's kino-dynamic constraints. The training procedure to extract subgoals from pedestrian trajectories is presented in Subsec III-B. We term the combination of VR and SE as the "Subgoal Recommender" (SR) network. In the final step, the extracted subgoal is passed to the MPC to calculate appropriate input commands satisfying collision avoidance constraints. This process is repeated continuously in a CL setting by adding an additional loss to restrict changes in the network's parameters that are essential to preserving performance in previously trained scenarios. We delve into the details of integrating the VR in a CL framework in Subsec III-C.

A. *Learning a Velocity Recommendation policy*

Our primary objective is to learn robot navigation policies that result in trajectories similar to pedestrians. Since velocity is the primary component that determines the nature of the trajectory, we propose to use IL [66] to predict what velocity a pedestrian would prefer to take in a given instant.

1) Velocity recommendation formulation: Given the current observation vector $\mathbf{o}_t$, we would like to determine the velocity $\mathbf{v}_{t+1}$ to be followed by the robot in the next time-step. We call this function π_{VR} and term the module as "Velocity Recommender".

$$\mathbf{v}_{t+1} = \pi_{VR}(\mathbf{o}_t)$$

2) Network architecture: An FCN with Linear Layers and Tanh activation is used, as shown in Figure 3.

The features used for the input $\mathbf{o}_t$ are:

$$\mathbf{o}_t = [\theta_t \ \mathbf{v}_t \ d_t^g \ \mathbf{d}_t^g \ r \ \mathbf{s}_t^{rel}]$$

where d_t^g, $\mathbf{d}_t^g$, r, $\mathbf{s}_t^{rel}$ denote the distance to goal, relative position vector to goal, the radius of robot and relative states of the other agents respectively. Hence, observation $\mathbf{o}_t$ correlates to the joint state vector s_t^{jn} introduced in Subsec. II-A.

Information about the robot's state $\mathbf{s}_t$ is contained in $[\theta_t \ \mathbf{v}_t \ d_t^g \ \mathbf{d}_t^g \ r]$ and the other agents' state information $\mathbf{z}_t$ is expressed relative to the robot's states resulting in the relative state for the i^{th} closest agent as:

$$\mathbf{s}_t^{rel_i} = [\mathbf{p}_t^{rel_i} \ \mathbf{v}_t^{rel_i} \ d_t^i]$$

where $\mathbf{p}_t^{rel_i}$, $\mathbf{v}_t^{rel_i}$ and d_t^i denote the relative position and velocity vectors of the i^{th} closest agent in robot's frame and the distance of the i^{th} closest agent from the robot. Hence, the combined relative state information vector for all the agents is given by:

$$\mathbf{s}_t^{rel} = [\mathbf{s}_t^{rel_1} \dots \mathbf{s}_t^{rel_n}]$$

where n is the total number of other agents in the scene.

$\mathbf{s}_t^{rel}$ is a crucial feature that warrants extra attention. It captures the information of the other agents in the environment and plays a major role in influencing human-robot interactions.

Normally there can be a variable number of agents in the robot's vicinity. Since it is impossible to represent the positions of a variable number of agents in a fixed dimensional vector, we limit the number of agents considered to be n_{max}. So, even if there are more than n_{max} other agents in the environment, information about the rest of the agents would be ignored by the policy due to this limitation.

3) Training procedure: We choose a simple BC algorithm to tackle the problem and assume that the model of the environment is not known and only samples of the expert's policy are given (Both state transition functions and the expert's policy function are unknown). Suppose there are n agents in a scene. In that case, expert demonstrations are a collection of trajectories denoted by $D = \{\tau_1, \tau_2 \ldots \tau_n\}$, where each trajectory τ_l is a set of state-action pairs for the l^{th} agent with $\tau_l = \{(\mathbf{p}_0^l, \mathbf{v}_0^l), (\mathbf{p}_1^l, \mathbf{v}_1^l) \ldots (\mathbf{p}_{t_l}^l, \mathbf{v}_{t_l}^l)\} \; \forall l \in [1, n]$. t_l is the time episode length of the l^{th} trajectory. We aim to learn the policy π_{VR}, such that it generates trajectories 'similar' to expert demonstrations D. Expert trajectory data consisting of position and velocity information of all the agents in a scene are collected, and the observation data $\mathbf{o}_t$ for each agent at every time step is computed using the robot's sensors. The tuple $(\mathbf{o}_t, \mathbf{v}_{t+1})$ serves as the training data for training the VR. It is essential to note that the training data consists of the observation and velocity that a pedestrian took in the following time step, not the current velocity. A Mean Squared Error (MSE) loss that encourages the network to predict velocities close to the ground truth is used to train the network as it is a regression problem. L_2 regularization loss is added in addition to prevent overfitting.

B. Extract subgoals from trajectories using Supervised Learning

We would like the robot to reach the velocity computed by the VR at each instant. However, due to robot's kinodynamic constraints, the robot may need to accelerate in a slightly different direction in order to reach the desired velocity at that instant. We learn this mapping between the desired velocity and the right acceleration direction in the form of subgoals. The subgoal points in the correct direction that the robot needs to "aim" to move towards to reach the desired velocity instantaneously. These subgoals are dynamically feasible as they account for the robot's dynamics already and can be used by any local planner to generate the appropriate robot input commands; we choose MPC to follow these subgoals.

Hence, the final objective is to extract subgoals from the predicted velocity commands that can be combined with MPC to generate safe input commands. We propose an algorithm to extract subgoals from rolled-out trajectories of agents following *any* policy. Typically humans have different dynamics from that of the robot, which means the internal subgoals that they might consider for navigation don't necessarily lead to the same trajectory when followed by the robot. We are interested in finding subgoals that the robot should consider that allows it to follow similar trajectories to the humans.

1) Subgoal extractor's formulation: Given the robot's state $\mathbf{s}_t$ and the desired velocity $\mathbf{v}_{t+1}$ for the next time-step, we would like to calculate which subgoal $\mathbf{p}_t^{sg}$ should be given to the MPC to calculate input commands that corresponds to the desired velocity.

We denote the function that computes the velocity by f_{MPC}. The velocity is calculated by: $\mathbf{v}_{t+1} = f_{MPC}(\mathbf{p}_t^{sg}, \mathbf{s}_t)$. Now, given the state at the current time step and velocity that MPC calculated for the next time-step, we would like to estimate which subgoal was considered in the MPC calculation. We call this function as π_{SG} and term this module as the "Subgoal Extractor" (SE). Like in [33], the difference between the global reference subgoal position and the current position, $\delta_t = (\mathbf{p}_t^{sg} - \mathbf{p}_t)$ is learned by the network.

$$\delta_t = \pi_{SG}(\mathbf{s}_t^{sg}, \mathbf{v}_{t+1}) \tag{6}$$

$$\mathbf{p}_t^{sg} = \delta_t + \mathbf{p}_t \tag{7}$$

By learning the mapping from velocity to subgoal, we learn the "inverse dynamics" of the robot that depends on the robot's forward dynamics and MPC algorithm. (The forward dynamics is the mapping from subgoals to velocity commands for the robot.) The features considered for the state vector $\mathbf{s}_t^{sg}$ are:

$$\mathbf{s}_t^{sg} = [\theta_t \; V_t] \tag{8}$$

The SE network completes the final part of the Subgoal Recommender's pipeline by recommending subgoals from the Velocity Recommender's velocity output for each observation.

2) Network Architecture: We aim to approximate π_{SG} using a Deep Neural Network (DNN). A FCN (Fully Connected Network) with 4 linear layers and Sigmoid activation function takes in the robot's state and desired velocity as input and returns the subgoal. The model architecture is shown in Fig. 4.

3) Training procedure: To generate the training data, we consider a scenario with no other dynamic agents in the environment and randomly sample subgoals that lie between the current position and the goal at each time-step and repeat it for different initial headings to achieve generalization. Each state-transition comprises of the tuple $(\mathbf{p}_t, \mathbf{s}_t^{sg}, \mathbf{v}_{t+1})$ and we collect $(\mathbf{s}_t^{sg}, \mathbf{v}_{t+1})$ for training the SE.

C. Adapting the policy to different scenarios

The second objective of our work is to adapt the trained socially compliant navigation policy to different scenarios without forgetting previously learned information. Thus, we train the VR in a CL setting.

1) Continual Learning Formulation: Let us consider k different scenarios $S = \{a_1 \ldots a_k\}$, where each scenario involves different social contexts. For instance, one scenario would consist of people preferring the left side, and a converse scenario would involve people taking the right side. For simplicity, we assume $k = 2$. $\{D_1, D_2\}$ correspond to expert trajectories obtained in scenarios a_1 and a_2 respectively. Initially, the robot must learn to imitate trajectories corresponding to D_1. The VR policy obtained as a result of training on D_1 is denoted by π_{VR}^1

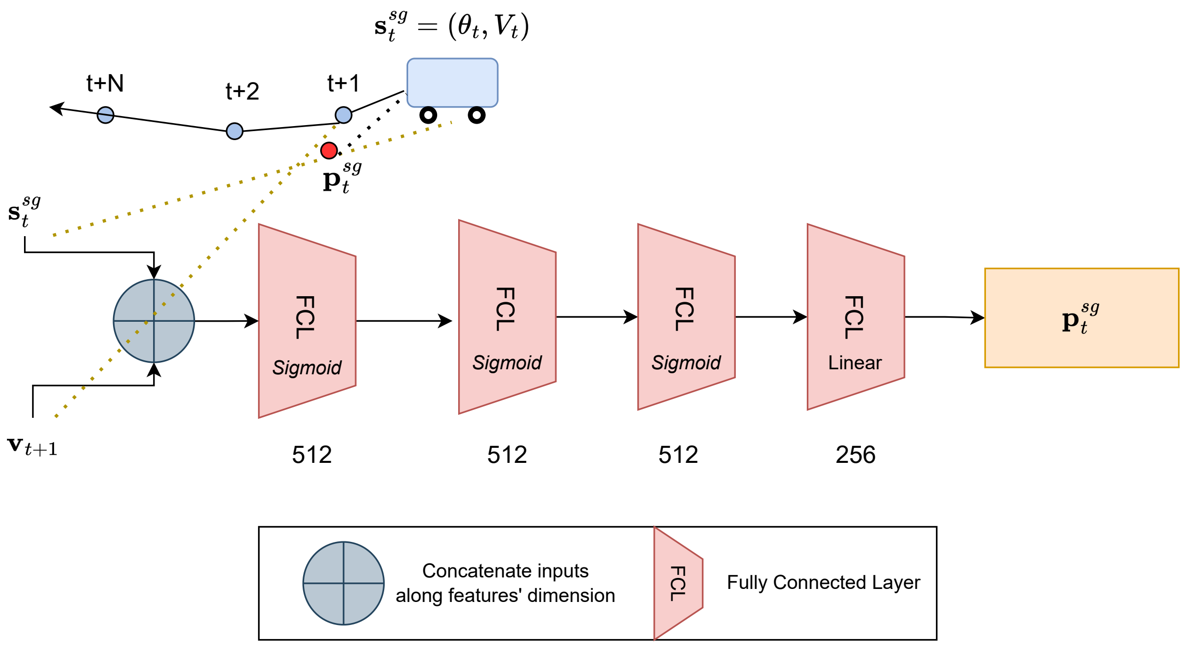


Fig. 4: Subgoal Extractor architecture diagram

and training on D_2 results in the policy π^2_{VR}. The objective is to adapt $\pi^1_{VR} \rightarrow \pi^{1,2}_{VR}$ such that the robot's resulting trajectories when following $\pi^{1,2}_{VR}$ in a_2 is similar to D_2, while being similar to D_1 in a_1. In general, if the robot encounters scenarios $a_1, a_2 \dots a_k$ consecutively with the corresponding expert demonstrations $D_1, D_2 \dots D_k$, we would like to arrive at a policy $\pi^{1,\dots,k}_{VR}$ that results in trajectories similar to D_j when deployed in scenario a_j $\forall j \in [1,k]$.

2) Training procedure: We employ Elastic Weight Consolidation (EWC) [52] to update the network parameters for learning the desired navigation behavior in new scenarios.

The training process is almost the same as in III-A3 with a minor modification in the loss function. In the original formulation, a Mean Squared Error (MSE) loss penalizing the L_2 norm between ground-truth labels and velocity predictions similar to Subsec. III-A3 is used for training the network. To incorporate CL, a regularization term is added to the original loss and the training is performed for each scenario consecutively. A coreset that contains a small subset of data from previously encountered scenarios is maintained and added during training to reduce catastrophic forgetting. The optimization step is performed using Stochastic Gradient Descent (SGD). For each scenario j, Fischer information matrix (F_j), which is an indication of the network parameters' (β_j) importance is stored after training in addition to β_j. Based on $F_{0:j-1}$ and $\beta_{0:j-1}$, the following regularization term is added to the loss function:

$$\mathcal{L}_{EWC}(\beta) = \sum_{j=0}^{k-1} \frac{\lambda}{2} F_j \left(\beta - \beta_j\right)^2$$

Where β is the current set of weights, k refers to the index of the current scenario, and λ is a hyperparameter used to balance between retaining old previous knowledge and learning new information. The complete loss function then looks as follows:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{MSE} + \mathcal{L}_{EWC} \\ &= ||\pi_{VR}(x_i) - y_i||_2^2 + \frac{\lambda}{2}\sum_{j=0}^{k-1} F_j ||\beta - \beta_j||_2^2\end{aligned}$$

where x_i and y_i denotes the input and the corresponding ground-truth label.

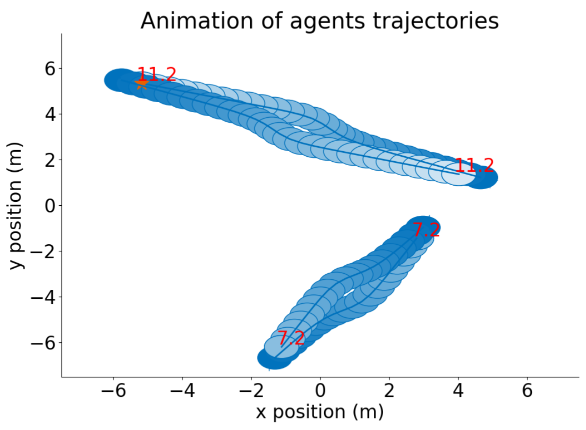
The pseudo-code for Continual Learning training is given in Algorithm 1.

Algorithm 1 Continual learning training

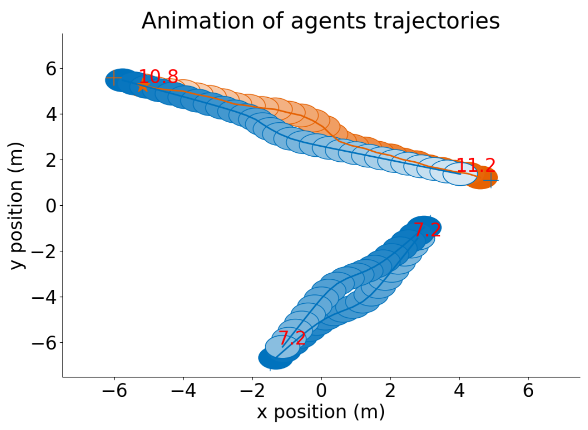
```
Input: $k, F_{0:k-1}, D_k, \beta_{0:k-1}, \pi^{k-1}_{VR}$.
Output: $\pi^k_{SR}, F_k$
1: $\beta \leftarrow InitializeParams()$
2: while $epochs < n_{epochs}$ do
3:     for $i = 0 \dots n_{mini\_batch}$ do
4:         $(x_i, y_i) \leftarrow Sample(D_k)$
5:         $\mathcal{L} \leftarrow ComputeL(x_i, y_i, \beta, \beta_{0:k-1})$
6:         $\beta \leftarrow SGDupdateStep(\mathcal{L})$
7:     end for
8: end while
9: $F_k \leftarrow CalculateFischer(\pi^k_{VR}, D_k)$
return $\{\pi^k_{VR}, F_k\}$
```

IV. Simulation Results

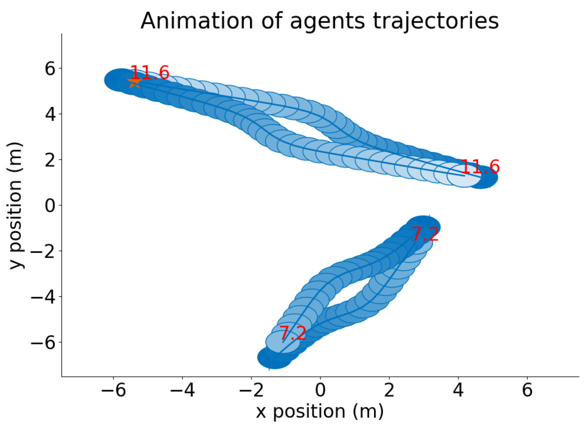
In this section, we evaluate each component of our proposed framework. The simulation settings used are detailed in Subsec



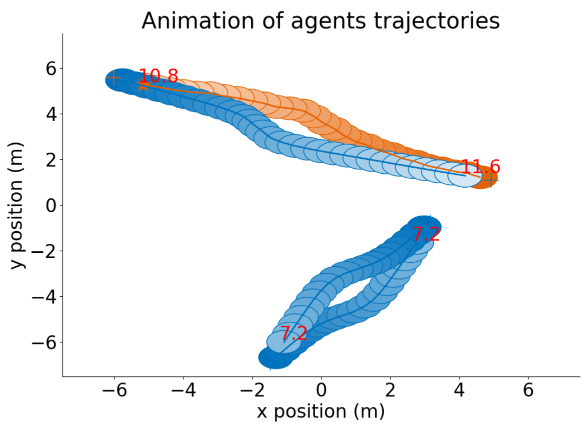
(a) SF (10)



(b) SESF-MPC (10)



(c) SF (25)



(d) SESF-MPC (25)

Fig. 5: Comparison between SF and SESF-MPC with other agents following SF (10) and SF (25) in PS scenario. As can be seen, the distance between the agents while crossing is higher for policies with higher SFF.

IV-A. The performance results for the SE, the SR and the CL are shown in Subsec. IV-B, IV-C and IV-D respectively. A key challenge in evaluating navigation policies trained on real pedestrian trajectories is that it isn't possible to compare the learned behavior against human behavior in reproducible scenarios. Although, there are pedestrian behavior models available, they do not represent all aspects of pedestrian navigation behavior. Hence, we initially consider a well known pedestrian model, namely the Social Forces (SF) model [36], to approximate pedestrian behavior and learn from the resulting trajectories. This allows a more extensive evaluation of our proposed method. We also test the method on real human data and show the results in the next section.

A. Simulation setup

We consider two variants of SF, where the Social Forces Factor (SFF) is increased from 20 to 25. SFF is a measure of how social we would like the agent to behave. For instance, a higher SFF leads to a larger distance maintained from the other agents. Given a maximum of 4 agents, the following scenarios are mainly considered in our simulations similar to [33]:

- **Pairwise Swap (PS)**: Initial positions are random. The agents are divided into pairs of two, and the task for each agent is to swap their positions with their counterpart.
- **Symmetric Swap (CS)**: Each agent's position is initialized in different parts of the quadrant in R^2 plane with equal distances from the origin and the agents from opposite quadrants need to swap their positions.

Since we are interested in evaluating the social compliance of the learned policy, we choose the following metrics as introduced in [4], [33] :

- Minimum distance: Distance between the ego-agent and the closest (by Euclidean distance) agent encountered.
- Trajectory smoothness: Mean change in heading between consecutive time-steps.
- Efficiency ratio: Ratio between the time taken to reach the goal if the ego-agent were to follow a straight line

path at the maximum velocity and the actual time taken to reach the goal.
- Number of collisions: Number of scenarios in which the distance between the center of the ego-agent and any other agent goes below a threshold value ϵ.
- Number of deadlocks: Number of scenarios in which the ego-agent gets stuck and fails to reach the goal.

B. Subgoal Extractor

In order to evaluate the performance of the SE, we would need the true subgoals to compare with the extracted subgoals. However, since the subgoal information for arbitrary policies is not available, we compare the ground truth trajectories with the trajectories resulting from following the extracted subgoals. When evaluating the performance of the SE in open-loop, the errors get accumulated. Hence, we use SF to get the next step's velocity at each time-step and pass it to the SE for calculating the subgoal. The extracted subgoal is then passed to the MPC which leads to a different robot state and the process is repeated in a closed loop fashion. We refer to this policy as "SESF-MPC" (Subgoals Extracted from SF with MPC).

The performance for all test scenarios are summarized in Table I. It is worth noting that the number of collisions have dropped down when using SESF-MPC compared to the ground truth, thus establishing the potential for combining learning based policies with MPC that lead to much safer trajectories. Further, we perform the 2-sample t-test between ground truth and SESF-MPC for each of the metrics to check if they are statistically close. It is worth noting that the collision avoidance of MPC is turned off for calculating the metrics to avoid influence of MPC in the calculations. We report the number of collisions and deadlocks for both cases of turning off and turning on the collision avoidance of MPC. The results of the t-test (Two-sided) is shown in Table II. The null hypothesis chosen is that the population means are equal for both the distributions. For a significance level of 0.05 ($\alpha = 0.05$), the corresponding threshold (t^*) is calculated to be 1.967 and 1.966 respectively for both the cases (The different values are due to different number of cases without collisions). The null hypothesis would be rejected if $|t| > |t^*|$. As can be seen from Table II, the calculated t values are within the threshold value for all the metrics in PS scenario implying that the ground truth and SESF-MPC policies result in trajectories that have similar characteristics with 95% confidence in that scenario. However except for the other agents' efficiency ratio, the rest of the metrics are not all within the threshold for CS scenario. For instance, the ego-agent's efficiency ratio have high magnitude t-values. This doesn't mean that the behavior is significantly different; it can partly be explained due to the fact that the variance for these metrics as seen in Table I are quite low. Narrow distributions tend to have very less overlap compared to wider distributions, thus leading to large differences in similarity metrics for the same difference in mean values. Since the efficiency ratio of the other agents is similar to SF for all the scenarios, we can infer that SESF-MPC doesn't disrupt the actions of other agents dramatically. It is safe to conclude that the behavior of SESF-MPC policy is close to that of SF policy's behavior. The plot corresponding to one of the test cases corresponding to PS scenario comparing SF and SESF-MPC policies are shown in Fig. 5. The difference in closest distance between the agents can be seen increasing from SFF 10 (Fig. 5(a) and 5(b)) to SFF 25 (Fig. 5(c) and 5(d)).

C. Subgoal Recommender

The SR policy which is a combination of VR and SE is obtained by following the training procedure mentioned in III-A3. In order to mitigate covariate shift, DAGGER [67] was used to improve the performance. The policy is tested against agents following SF policy on 200 random scenarios not seen during training and compared with the following baseline policies:

- SF (Ground truth)
- Goal Oriented - MPC (GO-MPC) : Trained against agents following SF(20) policy.
- Reciprocal Velocity Obstacles (RVO)

The resulting trajectories of all the policies for one of the scenarios is shown in Fig 6. The trajectory displayed by the agent following SR looks similar to that of the agent following SF for CS scenario with SFF 20. The performance metrics for all the test scenarios is tabulated in Table III. In order to verify that the observed collision avoidance behavior is not due to MPC, the collision avoidance was turned off for the plot as well as for calculating the metrics. We report the number of collisions and deadlocks both with and without collision avoidance of MPC. GO-MPC has the lowest number of collisions/deadlocks with collision avoidance. The number of collisions/deadlocks for SR is comparable to GO-MPC without collision avoidance. Further, in order to better compare the results, we repeat the t-test as explained in Subsec. IV-B. Table IV comprises of the comparison of the t-values for evaluating the trajectories of all the policies considered. Overall, almost all the metrics have t-values within the threshold for SR policy. It is not fair to compare the t-values for SR policy with RVO as they are not meant to imitate the SF policy unlike SR; we show them only for sake of completeness. Comparison between t-values of GO-MPC and SR is valid as GO-MPC was trained against agents following SF policy.

SR policy has t-values within the threshold for trajectory smoothness and other agents' efficiency ratio and is closer to SF policy's behavior for these metrics. GO-MPC has t-values within the threshold for the minimum distance kept, but SR keeps a higher distance than GO-MPC which makes our policy's behavior relatively more comfortable for pedestrians. Further, GO-MPC results in more efficient trajectories compared to SR with almost similar efficiency ratio values to that of RVO, which has highly efficient trajectories as it approaches the goal aggressively. This suggests that the GO-MPC agent has learned to exploit the SF agent's socially compliant behavior. This also can be inferred from the low trajectory smoothness values which suggests that the GO-MPC agent doesn't make as many turns compared to SR

Policy	Training and test scenarios	Other agents' policy	Min. distance (m)	Trajectory smoothness (rad)	Ego agent's efficiency ratio	Other agents' efficiency ratio	# Collisions / # Deadlocks (Without MPC)	# Collisions / # Deadlocks (With MPC)
Ground truth	PS	SF (20)	1.387 ± 0.146	0.630 ± 0.402	0.679 ± 0.253	0.675 ± 0.160	15 / 0	-
SESF-MPC (20)			1.374 ± 0.193	0.558 ± 0.369	0.711 ± 0.254	0.679 ± 0.158	10 / 0	5 / 0
Ground truth		SF (25)	1.469 ± 0.164	0.519 ± 0.331	0.672 ± 0.253	0.668 ± 0.161	12 / 0	-
SESF-MPC (25)			1.462 ± 0.161	0.537 ± 0.352	0.704 ± 0.256	0.664 ± 0.162	3 / 0	2 / 0
Ground truth	CS	SF (20)	1.415 ± 0.158	0.539 ± 0.147	0.423 ± 0.048	0.342 ± 0.050	16 / 0	-
SESF-MPC (20)			1.379 ± 0.163	0.515 ± 0.148	0.459 ± 0.019	0.341 ± 0.049	14 / 0	6 / 0
Ground truth		SF (25)	1.498 ± 0.170	0.583 ± 0.171	0.415 ± 0.052	0.338 ± 0.05	9 / 0	-
SESF-MPC (25)			1.466 ± 0.179	0.458± 0.132	0.455 ± 0.022	0.337 ± 0.050	11 / 0	7 / 0

TABLE I: Policy performance statistics with other agents (3) following SF policy in simulation. SESF-MPC policy is both trained and evaluated in PS and CS scenarios respectively.

Policy	Scenario	Min. distance t-value	Trajectory smoothness t-value	Ego agent's efficiency ratio t-value	Other agents' efficiency ratio t-value	$t^*(\alpha = 0.05)$
SESF-MPC (20)	PS	**0.717**	**1.786**	**-1.197**	**-0.235**	1.967
SESF-MPC (25)		**0.375**	**-0.493**	**-1.201**	**0.180**	1.966
SESF-MPC (20)	CS	2.092	**1.549**	-9.164	**0.193**	1.967
SESF-MPC (25)		**1.778**	7.943	-9.645	**0.176**	1.966

TABLE II: Subgoal Extractor: t values for calculated metrics by comparing SESF-MPC (20) and SESF-MPC (25) with SF (20) and SF(25) respectively in PS and CS scenarios.

Policy	Other agents' policy	Min. distance (m)	Trajectory smoothness (rad)	Ego agent's efficiency ratio	Other agents' efficiency ratio	# Collisions / # Deadlocks (Without MPC)	# Collisions / # Deadlocks (With MPC)
Ground truth	SF (20)	1.396 ± 0.139	0.648 ± 0.439	0.674 ± 0.252	0.674 ± 0.160	17 / 0	-
RVO		1.135 ± 0.194	0.518 ± 0.368	0.739 ± 0.243	0.678 ± 0.155	9 / 4	-
GO-MPC (20)		1.411 ± 0.353	0.505 ± 0.335	0.738 ± 0.240	0.681 ± 0.156	20 / 0	6 / 1
SR (20)		1.539 ± 0.482	0.606 ± 0.397	0.725 ± 0.258	0.677 ± 0.162	18 / 0	8 / 1
Ground truth	SF (25)	1.479 ± 0.155	0.535 ± 0.379	0.666 ± 0.249	0.669 ± 0.162	15 / 0	-
RVO		1.179 ± 0.193	0.335 ± 0.241	0.739 ± 0.241	0.665 ± 0.159	9 / 0	-
GO-MPC (20)		1.466 ± 0.333	0.342 ± 0.238	0.741 ± 0.243	0.675 ± 0.159	13 / 0	5 / 1
SR (20)		1.567 ± 0.460	0.516 ± 0.342	0.731 ± 0.252	0.670 ± 0.161	16 / 0	9 / 5

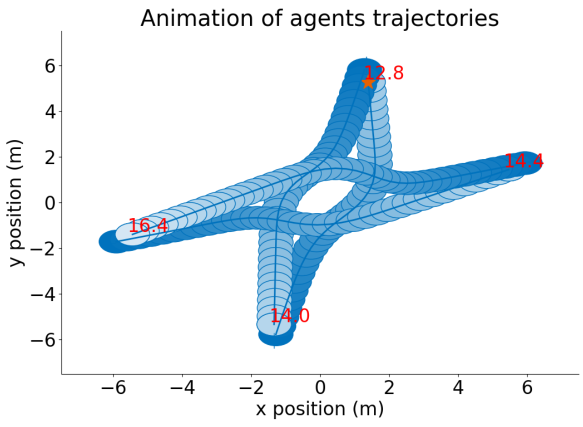
TABLE III: Policy performance statistics of various policies with baseline policies like RVO and GO-MPC along with our SR policy and other agents (3) following SF(20) and SF(25) policies in PS scenario.

Policy	Other agents' policy	Min. distance t-value	Trajectory smoothness t-value	Ego agent's efficiency ratio t-value	Other agents' efficiency ratio t-value	$t^*(\alpha = 0.05)$
RVO	SF (20)	14.518	3.002	-2.472	**-0.206**	1.967
GO-MPC (20)		**-0.526**	3.344	-2.371	**-0.423**	
SR (20)		-3.784	**0.941**	**-1.859**	**-0.155**	
RVO	SF (25)	16.193	6.014	-2.809	**0.263**	1.967
GO-MPC (20)		**0.456**	5.748	-2.872	**-0.351**	
SR (20)		-2.416	**0.505**	-2.427	**-0.065**	

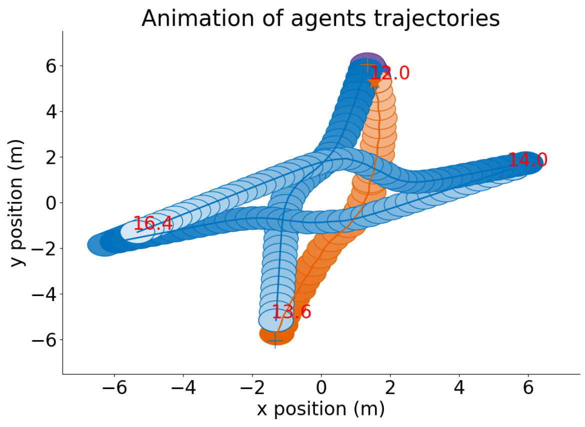
TABLE IV: t values for calculated metrics comparing some baseline policies like RVO and GO-MPC along with our SR policy in PS scenario.

agent thus making the other agents swerve around instead. The exploitation behavior of GO-MPC can be seen in Fig. 13. However, the efficiency ratio of other agents' is similar for all the policies which means that the time taken for the other agents to reach the goal is not very different from the ground truth.

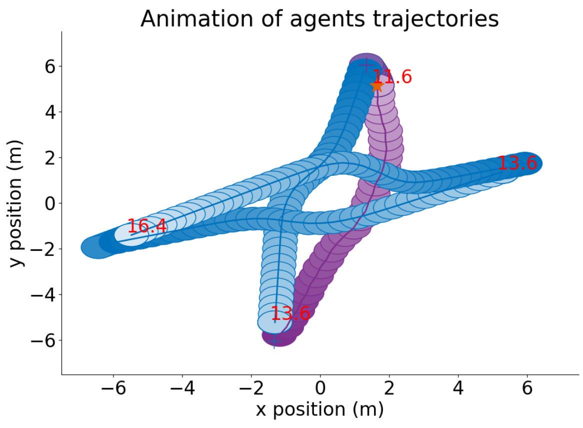
Overall, SR results in trajectories that are very close to SF in terms of smoothness and a more conservative behavior as



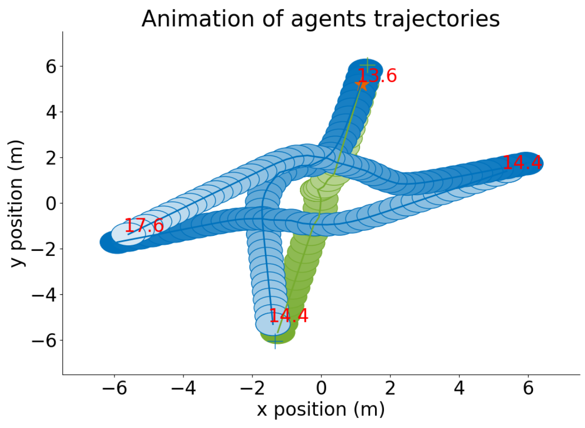
(a) SF (20)



(b) SR (20)



(c) GO-MPC



(d) RVO

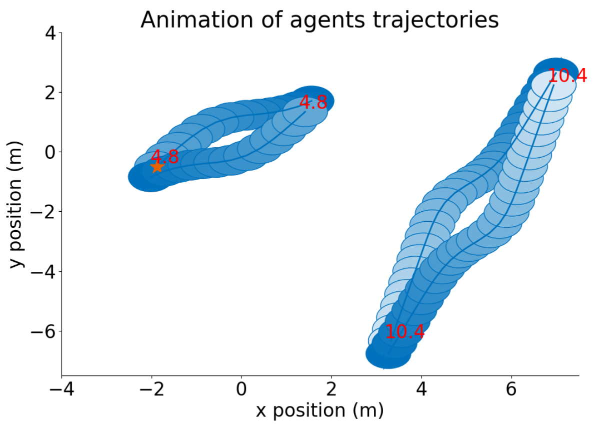
Fig. 6: Simulation with other agents following SF(20) in CS scenario. The SR(20) policy means it was trained from observing agents following SF (20) policy. We deploy SR(20) trained in PS scenario on CS scenario; clearly the policy generalizes well.

opposed to GO-MPC which exploits the other agents' behavior and RVO which has no regard for proxemics.

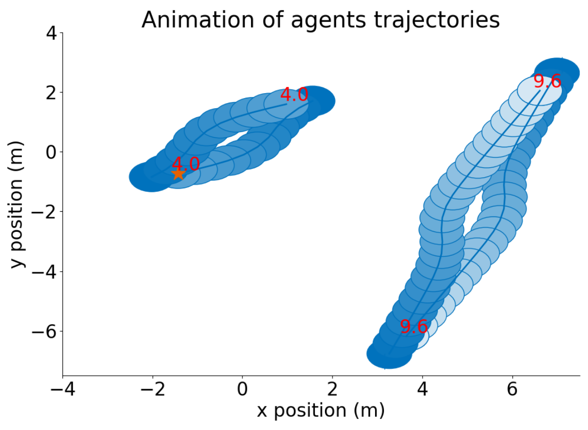
D. *Continual Learning*

In Subsec. IV-C, we demonstrated how SR policy performs in test scenarios that belong to the same data distribution from which the training data were sampled. Now, we would like to investigate the effect on SR policy's performance on test scenarios in which the agents display different degrees of social compliance; in particular, different side preferences. Agents following SF (20) policy tend to prefer the right hand side, while agents following SF (-20) prefer the left side. The resulting trajectories from different side preferences are shown in Fig 7. We consider a SR policy that is initially trained by observing agents following SF (20) and then adapted to behave similar to agents following SF (-20) policy using the CL algorithm described in Alg. 1. We demonstrate the effects of both Sequential and Continual Learning of SR (20) and SR (-20) policies with the same model in Fig. 8. In Sequential Learning, the model is trained using data from the current social context without applying EWC loss. The first learning curve corresponding to Sequential Learning in Fig.8(a) clearly shows an increase in the validation and test loss for SR (20) when the model starts to learn from data corresponding to SR (-20); this is the essence of Catastrophic Forgetting. For CL, the EWC loss is applied in combination with a coreset from the previous social context while adapting the model to learn the current social context The second learning curve in Fig.8(b) corresponding to CL displays a saturation in the validation and test loss of SR (20), implying that the data corresponding to SR(-20) is learned in a continual fashion.

The performance of SR policy trained to prefer the right side and then adapted to prefer the left side applying CL in comparison with SR policies trained to only prefer the right or left side is shown in Table V. Since the focus here is on how well the policy adapts to different side preferences, we compare the number of collisions and deadlocks and number of wrong sides. The number of wrong sides refer to cases when

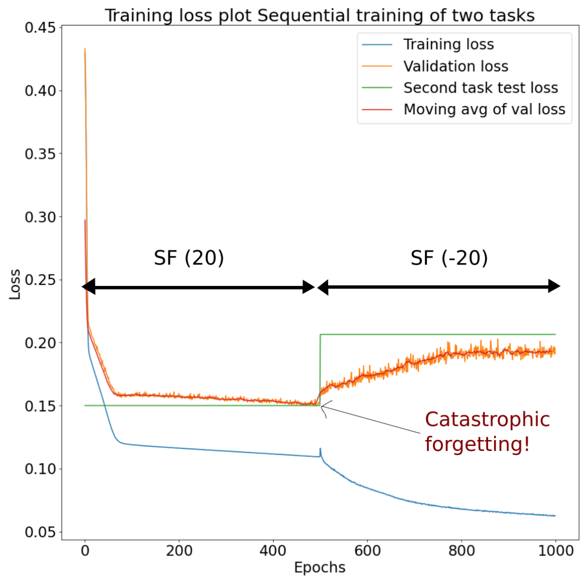


(a) SF (20) - Right

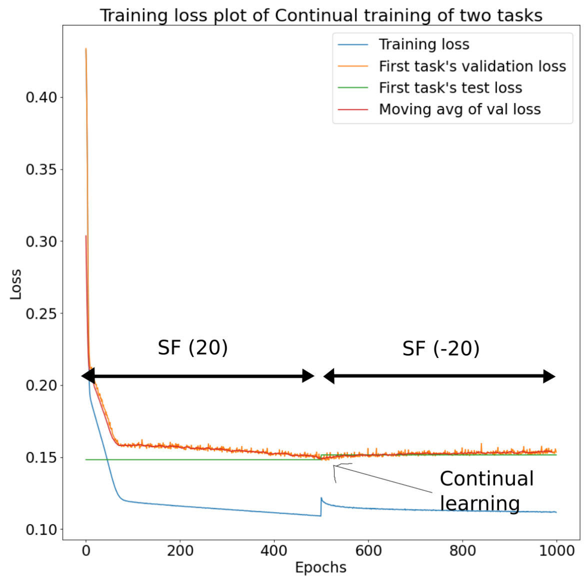


(b) SF (-20) - Left

Fig. 7: Simulation with all agents following SF(20) and SF (-20) in PS scenario. The agents prefer to keep on the right side in SF(20), while SF(-20) encourages agents to prefer the left. Increase in lightness of the agent's colour is further into the future.



(a) SR (20) , SR (-20)



(b) CL(20)(-20)

Fig. 8: Learning curves containing loss plots for Sequential Learning (top) and Continual learning (bottom). The training loss corresponds to the task learned currently, while the validation and test losses always corresponds to the first task learned to observe how previously learned information is retained.

the robot doesn't take the side preferred by the other agent; it is counted only for the cases where the agent manages to reach the goal. Policies trained to make the robot prefer one side are tested against an agent preferring the opposite side. For instance, SR (20) where the robot prefers right side is tested against an agent following SF (-20) preferring the left side and vice-versa.

It is immediately obvious that having different side preferences than the other agents in the environment, leads to a sharp decline in performance as the number of collisions increase drastically, especially in the case of SR (20) deployed amongst an agent following SF (-20). Even MPC is not able to avoid the collisions since the SR policy constantly recommends subgoals that lead to trajectories inadvertently crashing into the very agent it's trying to stay away from. It is clear from Table V that once CL is applied, the number of collisions drop down and the robot is able to keep the correct side for majority of the test cases.

We show an interesting case in Figure 9. The collision avoidance is turned on here for better inference. The agent in Fig. 9(a) following SR (20) collides with the agent following SF (-20) and the agent following CL(20)(-20) in Fig. 9(b) is able to keep left throughout since it was trained to keep to the left side in it's latest training. In fact, the agent following CL(20)(-20) in Fig 9(d) initially tilts towards the left side against the agent following SF (20), but on observing that the other agent prefers right, it is able to steer away on getting closer. Thus, we are able to illustrate how the agent following CL(20)(-20) is still able to adapt to keep to the left side while not forgetting the previously learned right side preference behavior.

Ego Policy	Other agent's Policy	# wrong sides	# Collisions / # Deadlocks (without MPC)	# Collisions / # Deadlocks (with MPC)
SR (20)	SF (20)	0	4 / 0	1 / 0
	SF (-20)	0	192 / 0	140 / 0
SR (-20)	SF (20)	17	173 / 0	85 / 1
	SF (-20)	0	15 / 0	12 / 0
SR (20 + -20)	SF (20)	13	40 / 0	40 / 0
	SF (-20)	0	48 / 0	48 / 0

TABLE V: Policy performance statistics of CL policy against another agent following SF (20) and SF (-20) policies in PS scenario. The performance of SR (20) and SR (-20) against SF (-20) and SF (20) respectively are also included for comparison.

V. Experimental validation

In section IV, the SR policy was trained based on rolled-out trajectories from agents following the SF policy in simulation. In this section, we demonstrate the performance of the policy trained on real pedestrian trajectories. We describe the experimental setup used for recording the data in Subsec. V-A, followed by performance results of the SR policy trained on the collected pedestrian data in Subsec. V-B and CL results in Subsec. V-C.

A. *Experimental setup*

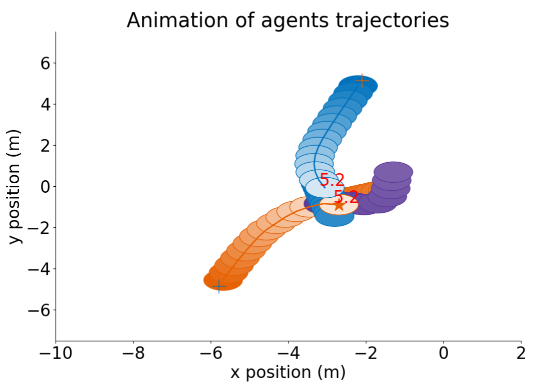
A Clearpath Jackal robot equipped with five RealSense cameras, an Ouster Lidar and an internal onboard computer running ROS Melodic is used in our experiments. The perception pipeline contains a Person Detection package using which we extract each pedestrian's trajectory data in the scene. A snapshot of the experimental setting with the Jackal recording pedestrian trajectory data is shown in Fig. 1. Data of two agents walking in multiple PS scenarios for both left and right preferring cases are recorded. The result of replaying one such recorded trajectory in the Gym Collision Avoidance environment is shown in Fig. 10.

B. *Subgoal Recommender from real pedestrian data*

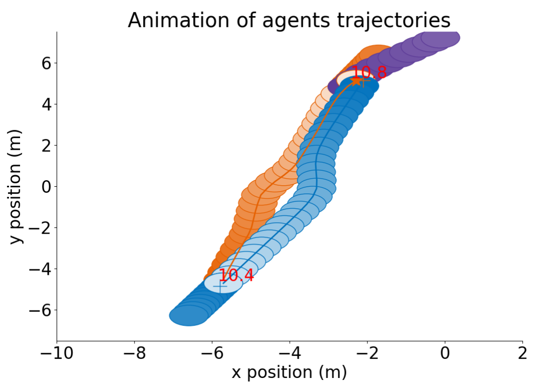
The recorded trajectories are used to train left and right preferring SR policies separately which we term as SR (left) and SR (right) respectively. Training data from the ETH dataset [68] is augmented to improve the training process. The trained policy is deployed against agents following SF policy with the same side preference. A plot showing the performance of both SR (right) and SR (left) is shown in Fig. 11. The performance metrics of the trained policies are shown in Table VI. The number of collisions is reasonably low for both the policies. SR (left) has relatively higher number of collisions and time-outs because the amount of training data collected for the left scenarios was lower than that of the right scenarios. Further as evidenced by the much higher minimum distance, the agent following SR (left) policy is more conservative from the beginning which compromises the ability to reach the goal perfectly.

C. *Continual learning of social preferences*

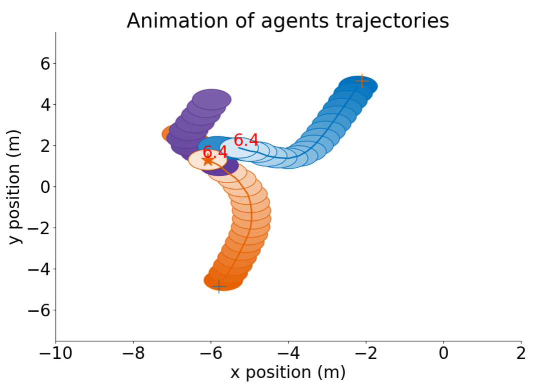
In Subsec. V-B, separate policies were trained for learning left and right preferring policies. However, we are interested in learning both right and left preferring behaviors based on other agents' preference. To demonstrate CL of different social preferences from real human data, we train the agent to learn a right preferring policy and then adapt it to keep to the left. The performance results of the CL(right)(left) policy tested against both right and left preferring agents is shown in Table VII. As expected, the performance of policies trained to prefer one side leads to a lot of collisions when tested against an agent preferring the opposite side. The continually learned policy leads to much lower collisions compared to the individually trained policies when tested against agents preferring the opposite side. However, the number of collisions is still not close to the performance of individual policies tested against agents preferring the same side. This can be attributed to the limitation in the network architecture and the trade-off between plasticity and elasticity of network parameters for retaining and learning new information by means of hyper-parameter λ. The results of the policy is shown in Fig. 12. As observed in Subsec. IV-D, the agent prefers the left side initially and then swerves to the right as seen in Fig. 12(a). The agent is able to keep left throughout against the left preferring agent as seen in Fig. 12(b), since it corresponds to the latest



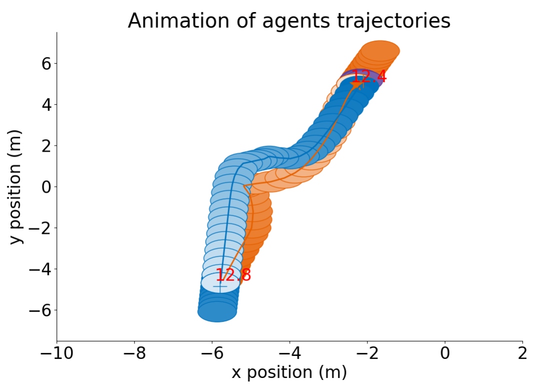
(a) SR (20) against SF(-20)



(b) CL(20)(-20) against SF(-20)

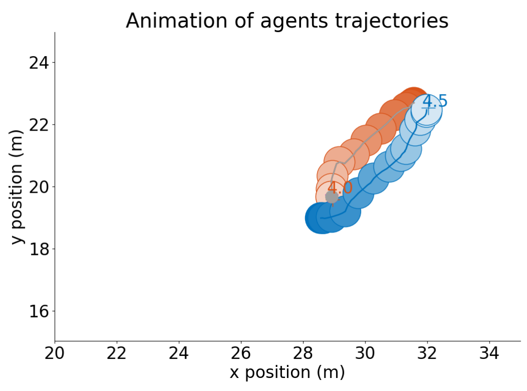


(c) SR (-20) against SF(20)

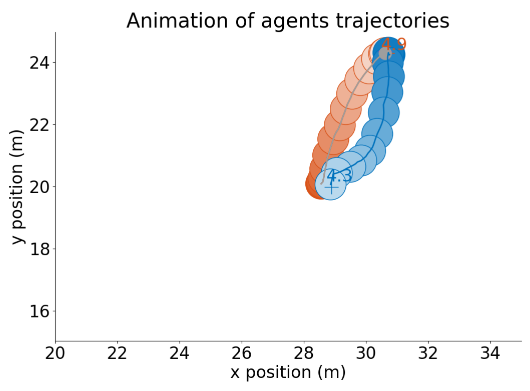


(d) CL(20)(-20) against SF(20)

Fig. 9: Simulation showing cases where deploying SR(20) (on top-left) and SR(-20) (on bottom-left) with other agents following SF(-20) and SF(20) leads to collisions. In contrast, when the policy is adapted using CL (on right column), the agent no longer collides on adapting to the side preferences.

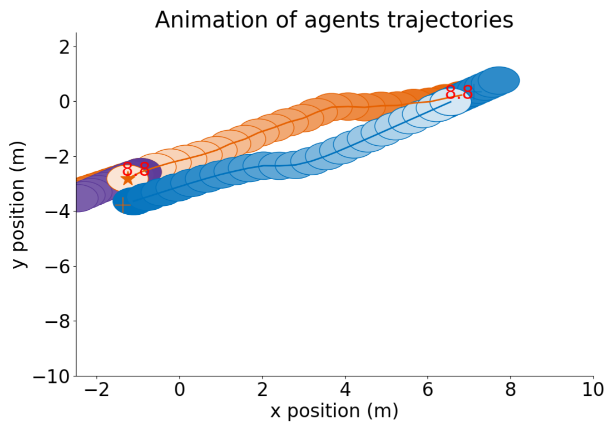


(a) SF (20)

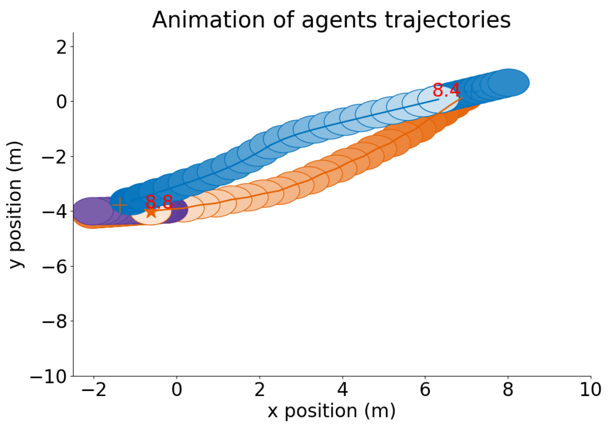


(b) SR (-20)

Fig. 10: Plot of agents' trajectories following right and left preferring policies of two people walking in TU Delft 3me Dept.



(a) SR (right)



(b) SR (left)

Fig. 11: Simulation with ego agent following SR(right) and SR (left) in PS scenario against agent following SF(20) and SF(-20) respectively.

Ego Policy	Other agents' Policy	Min. distance (m)	Trajectory smoothness (rad)	Ego agent's efficiency ratio	Other agents' efficiency ratio	# Collisions / # Deadlocks (without MPC)	# Collisions / # Deadlocks (with MPC)
SR (right)	SF (20)	1.229 ± 0.201	0.601 ± 0.420	0.738 ± 0.243	0.705 ± 0.243	13 / 0	12 / 0
SR (left)	SF (-20)	1.798 ± 0.554	0.615 ± 0.381	0.583 ± 0.296	0.723 ± 0.245	26 / 5	1 / 21

TABLE VI: Policy performance statistics of SR (right) and SR (left) policies trained with real pedestrian data and tested against an agent following SF policy with the same side preference in PS scenario.

learned behavior. Thus, we are able to validate our proposed framework's performance on real data.

Ego Policy	Other agents' Policy	# wrong directions	# Collisions / # Deadlocks (without MPC)	# Collisions / # Deadlocks (with MPC)
CL(right)(left)	SF (20)	21	58 / 1	19 / 15
	SF (-20)	2	66 / 1	13 / 7
SR (right)	SF (20)	0	13 / 0	12 / 0
	SF (-20)	1	168 / 0	54 / 1
SR (left)	SF (20)	56	114 / 0	8 / 32
	SF (-20)	0	26 / 5	1 / 21

TABLE VII: Policy performance statistics of CL(right)(left) policy trained with real pedestrian data with right and left preferences using CL and tested against an agent following SF policy with the both side preferences in PS scenario. The results of SR(right) and SR(left) tested against an agent following the opposite side preference is also shown for comparison.

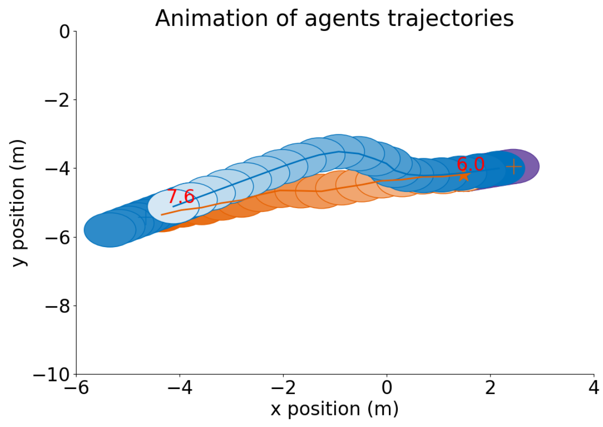
VI. Conclusion

This work introduced a methodology to learn a Subgoal Recommender policy that provides intermediate subgoals for the robot by continuously observing pedestrian trajectories in the environment. The learned Subgoal Recommender is used to guide an optimization-based local planner to generate trajectories with fewer collisions while being socially compliant. We employed Behavior Cloning to learn subgoal recommendations that are kinodynamically feasible from observations of pedestrian trajectories and combined it with MPC to calculate input commands to the robot that satisfy kinodynamic and collision avoidance constraints. Our policy is shown to generate trajectories with similar smoothness and efficiency as that of the Social Forces policy, whose behavior was learned with 95% confidence verified by t-tests. In addition to that, our policy maintains more minimum distance compared to the Social Forces policy, thus respecting pedestrians' comfort. Further, the policy can adapt to the environment when other agents display varying side preferences without forgetting the desired behavior for previously encountered social contexts, which was tested by changing the side preference of the other agents.

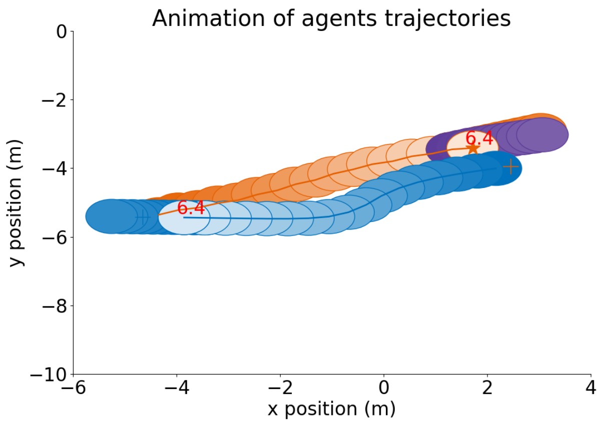
VII. Limitations and Future work

The following points list some limitations and suggestions on overcoming them:

- Our work only considers human-human interactions and does not incorporate environmental constraints. Information about the occupancy grid map can be included in the input features to allow the network to recommend subgoals that do not lead the robot to bump into walls, for example.
- Since we used an FCN, only a fixed number of input features could be used, due to which the number of agents considered was limited to n_{max}. We can account for all the agents in the environment by having an attention



(a) CL(right)(left) against
SF (20)



(b) CL(right)(left) against
SF (-20)

Fig. 12: Simulation with ego agent being able to display both left and right preferring navigation behavior with the same policy.

layer to reduce variable-sized input features to a fixed-size feature vector. This way, the network can learn to focus on reacting to people who are most likely to cross paths with the robot.

- The MPC planner uses a constant velocity model for predicting the pedestrian's velocity. Though the CV model is shown to have excellent performance over various learning methods [69], it still needs improvement. Hence, we cannot necessarily avoid colliding with pedestrians if they walk erratically.
- We chose BC combined with DAGGER to perform IL in simulations. However, since the expert policy is not available for the actual data, we can try out Adversarial algorithms like GAIL [70] to mitigate the accumulation of errors due to covariate shift [51].
- An RNN architecture that can keep track of the history of observations can be used to experiment if it is possible to infer the side preference of the other agents early on to adapt the robot's behavior as required in the beginning.
- We assumed that the information on when the distribution shift happens in the environment is given to us. In other words, we assume that the robot is aware when pedestrians exhibit a different level of social compliance, and the policy needs to be adapted. This assumption may not hold in reality, and other metrics like perceived comfort that measure the drop in performance should be used to inform the pipeline to retrain the policy.
- The chosen social scenarios are opposite, meaning the agent is forced to unlearn previously acquired information to adapt to the new situation. It would be interesting to test the CL policy on tasks that do not have conflicting requirements.

REFERENCES

[1] P. Kurup and K. Liu, "Telepresence robot with autonomous navigation and virtual reality: Demo abstract," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, pp. 316–317.

[2] "Mobile robot transports sterile goods in hospital." [Online]. Available: https://ifr.org/ifr-press-releases/news/mobile-robot-transports-sterile-goods-in-hospital

[3] G. Fragapane, H.-H. Hvolby, F. Sgarbossa, and J. O. Strandhagen, "Autonomous mobile robots in hospital logistics," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2020, pp. 672–679.

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

[5] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[6] T. Kruse, P. Basili, S. Glasauer, and A. Kirsch, "Legible robot navigation in the proximity of moving humans," pp. 83–88, 2012.

[7] C. Rösmann, M. Oeljeklaus, F. Hoffmann, and T. Bertram, "Online trajectory prediction and planning for social robot navigation," pp. 1255–1260, 2017.

[8] A. J. Sathyamoorthy, U. Patel, M. Paul, N. K. S. Kumar, Y. Savle, and D. Manocha, "Comet: modeling group cohesion for socially compliant robot navigation in crowded scenes," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1008–1015, 2021.

[9] H. Khambhaita and R. Alami, "Viewing robot navigation in human environment as a cooperative activity," in *Robotics Research*. Springer, 2020, pp. 285–300.

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

[11] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2096–2101.

[12] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[13] P. T. Singamaneni, A. Favier, and R. Alami, "Human-aware navigation planner for diverse human-robot interaction contexts," in *2021 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 5817–5824.

[14] P. Teja S. and R. Alami, "Hateb-2: Reactive planning and decision making in human-robot co-navigation," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020, pp. 179–186.

[15] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022.

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

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

[18] Ó. Gil and A. Sanfeliu, "Effects of a social force model reward in robot navigation based on deep reinforcement learning," in *Iberian Robotics conference*. Springer, 2019, pp. 213–224.

[19] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3517–3524.

[20] J. Bi, T. Xiao, Q. Sun, and C. Xu, "Navigation by imitation in a pedestrian-rich environment," *arXiv preprint arXiv:1811.00506*, 2018.

[21] M. Fahad, G. Yang, and Y. Guo, "Learning human navigation behavior using measured human trajectories in crowded spaces," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 154–11 160.

[22] Q. Fang, X. Xu, X. Wang, and Y. Zeng, "Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning," *CAAI Transactions on Intelligence Technology*, 2021.

[23] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne, "Deep imitation learning for 3d navigation tasks," *Neural computing and applications*, vol. 29, no. 7, pp. 389–404, 2018.

[24] A. Konar, B. H. Baghi, and G. Dudek, "Learning goal conditioned socially compliant navigation from demonstration using risk-based features," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 651–658, 2021.

[25] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[26] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[27] C.-E. Tsai and J. Oh, "A generative approach for socially compliant navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2160–2166.

[28] S. Wei, X. Chen, X. Zhang, and C. Qi, "Towards safe and socially compliant map-less navigation by leveraging prior demonstrations," in *International Conference on Intelligent Robotics and Applications*. Springer, 2020, pp. 133–145.

[29] B. Xiong, F. Wang, C. Yu, F. Qiao, Y. Yang, Q. Wei, and X.-J. Liu, "Learning safety-aware policy with imitation learning for context-adaptive navigation," 2019.

[30] C. Wang, X. Liu, X. Yang, F. Hu, A. Jiang, and C. Yang, "Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy," *Applied Sciences*, vol. 8, no. 2, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/2/231

[31] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[32] D. Limon, J. Calliess, and J. M. Maciejowski, "Learning-based nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7769–7776, 2017.

[33] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation among pedestrians." [Online]. Available: https://arxiv.org/abs/2102.13073

[34] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *arXiv preprint arXiv:1811.01848*, 2018.

[35] F. Farshidian, D. Hoeller, and M. Hutter, "Deep value model predictive control," *arXiv preprint arXiv:1910.03358*, 2019.

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

[37] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[38] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[39] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "Brvo: Predicting pedestrian trajectories using velocity-space reasoning," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 201–217, 2015.

[40] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.

[41] P. A. Vasconcelos, H. N. Pereira, D. G. Macharet, and E. R. Nascimento, "Socially acceptable robot navigation in the presence of humans," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*. IEEE, 2015, pp. 222–227.

[42] Y. Wang, D. Zhang, J. Wang, Z. Chen, Y. Li, Y. Wang, and R. Xiong, "Imitation learning of hierarchical driving model: From continuous intention to continuous trajectory," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2477–2484, 2021.

[43] P. Cai, Y. Sun, H. Wang, and M. Liu, "Vtgnet: A vision-based trajectory generation network for autonomous vehicles in urban environments," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 3, pp. 419–429, 2020.

[44] L. Qin, Z. Huang, C. Zhang, H. Guo, M. Ang, and D. Rus, "Deep imitation learning for autonomous navigation in dynamic pedestrian environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4108–4115.

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

[46] B. H. Baghi and G. Dudek, "Sample efficient social navigation using inverse reinforcement learning," 2021.

[47] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[48] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," vol. 30, 2017.

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

[50] M. Sayed-Mouchaweh and E. Lughofer, *Learning in non-stationary environments: methods and applications*. Springer Science & Business Media, 2012.

[51] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[52] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, mar 2017.

[53] K. Shaheen, M. A. Hanif, O. Hasan, and M. Shafique, "Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 1, pp. 1–32, 2022.

[54] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[55] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances in neural information processing systems*, vol. 30, 2017.
[56] R. Kemker and C. Kanan, "Fearnet: Brain-inspired model for incremental learning," 2018.
[57] H.-R. Wei, S. Huang, R. Wang, X. Dai, and J. Chen, "Online distilling from checkpoints for neural machine translation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 1932–1941.
[58] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.
[59] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1606.04671
[60] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," 2017. [Online]. Available: https://arxiv.org/abs/1708.01547
[61] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 254–11 263.
[62] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 203–10 209.
[63] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[64] M. Sadeghi Lahijani, T. Islam, A. Srinivasan, and S. Namilae, "Constrained linear movement model (calm): Simulation of passenger movement in airplanes," *PLoS one*, vol. 15, no. 3, p. e0229690, 2020.
[65] D. DeVon and T. Bretl, "Kinematic and dynamic control of a wheeled mobile robot," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 4065–4070.
[66] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018. [Online]. Available: https://arxiv.org/abs/1805.01954
[67] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.
[68] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 261–268.
[69] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1696–1703, 2020.
[70] J. Ho and S. Ermon, "Generative adversarial imitation learning," 2016.

VIII. Appendix

The following table shows values of some important parameters introduced in the paper.

ϵ (m)	n_{max}	λ	Learning rate (α)	Weight decay
0.05	9	100	5×10^{-4}	1×10^{-3}

TABLE VIII: Important hyperparameters

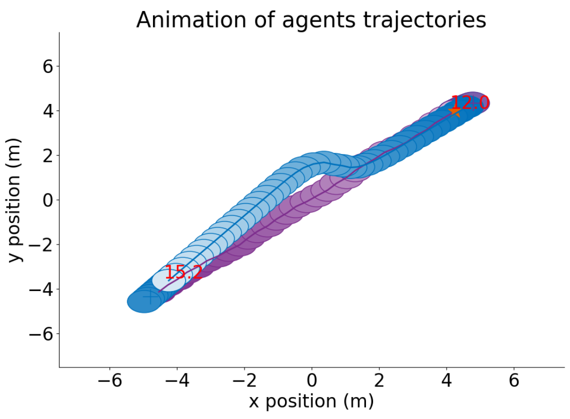


Fig. 13: GOMPC exploiting the social compliance of SF agent and goes straight without regard for comfort.