

# SHINE: Social Homology Identification for Navigation in Crowded Environments

Journal Title  
XX(X):1–18  
©The Author(s) 2024  
Reprints and permission:  
sagepub.co.uk/journalsPermissions.nav  
DOI: 10.1177/ToBeAssigned  
www.sagepub.com/

SAGE

Diego Martinez-Baselga<sup>1</sup>, Oscar de Groot<sup>2</sup>, Luzia Knoedler<sup>2</sup>, Luis Riazuelo<sup>1</sup>, Javier Alonso-Mora<sup>2</sup>, Luis Montano<sup>1</sup>

## Abstract

Navigating mobile robots in social environments remains a challenging task due to the intricacies of human-robot interactions. Most of the motion planners designed for crowded and dynamic environments focus on choosing the best velocity to reach the goal while avoiding collisions, but do not explicitly consider the high-level navigation behavior (avoiding through the left or right side, letting others pass or passing before others, etc.). In this work, we present a novel motion planner that incorporates topology distinct paths representing diverse navigation strategies around humans. The planner selects the topology class that imitates human behavior the best using a deep neural network model trained on real-world human motion data, ensuring socially intelligent and contextually aware navigation. Our system refines the chosen path through an optimization-based local planner in real time, ensuring seamless adherence to desired social behaviors. In this way, we decouple perception and local planning from the decision-making process. We evaluate the prediction accuracy of the network with real-world data. In addition, we assess the navigation capabilities in both simulation and a real-world platform, comparing it with other state-of-the-art planners. We demonstrate that our planner exhibits socially desirable behaviors and shows a smooth and remarkable performance.

## Keywords

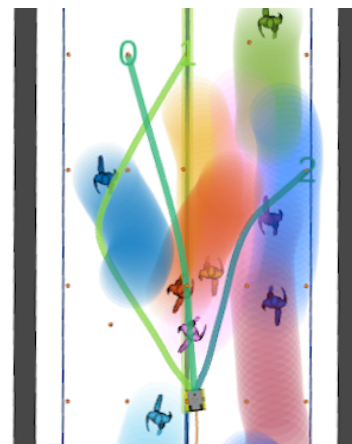
Human-Aware Motion Planning, Motion and Path Planning, Collision Avoidance, Integrated Planning and Learning

## 1 Introduction

Some of the most important applications of mobile robots involve moving alongside humans. However, their seamless integration into social settings presents notable challenges. These challenges include aspects such as safety, efficiency, smooth interactions, adherence to social conventions, and gaining human acceptance. Addressing them is essential for the successful integration of robots into social spaces.

Motion planning in dynamic environments involves global and local planning. Traditional global planners compute a global plan that only considers static obstacles and a local planner tries to follow it. Thus, the local planner may find itself stuck in locally optimal plans that overlook the presence of dynamic obstacles beyond its planning horizon. This can lead to different dynamic behaviors that were not considered by the initial plan, such as passing before obstacles or letting them pass.

In addition, social navigation poses the challenge of selecting the desired robot dynamic behavior. While manually defining a cost based on heuristics is an option, evaluating social norms and adapting to every possible situation is a very complex problem. For example, in Figure 1, the robot faces a very dense scenario where it has to make a high-level decision, avoid and cooperate with humans to advance in a corridor. It may wait for the humans that are in front of it to move away or navigate through the left or right sides, dealing with the humans that it encounters there. In other scenarios, those kind of decisions could include aspects such as avoiding uncomfortable situations, not blocking other's way or understanding preferences.



**Figure 1.** A dense corridor scenario, where the robot may follow one of three high level behaviors to progress. The behaviors are represented with three possible paths, and the future trajectories of humans with semi transparent shadows.

This work aims to develop a motion planner, which we name **SHINE** (Social Homology Identification for

<sup>1</sup>University of Zaragoza, Spain

<sup>2</sup>Delft University of Technology, Netherlands

### Corresponding author:

Diego Martinez-Baselga, Robotics, Computer Vision and Artificial Intelligence Group, Institute of Engineering Research (I3A), University of Zaragoza, 50018 Zaragoza, Spain.

Email: diegomartinez@unizar.es

Navigation in crowded Environments), that uses high-level navigation decisions learned from humans for navigating in crowded environments. Out of the different trajectories that a robot can follow to reach its goal, some exhibit similar topological characteristics. High-level decisions are understood as the possible navigation topology classes of the trajectories with respect to the surrounding humans of the environment. They also represent different feasible local optimal routes to a goal when considering robotic navigation as an optimization problem. The proposed planner first computes topology distinct guidance trajectories using a Visibility-Probabilistic Road Maps (PRM) method (de Groot et al. 2023), and chooses one based on a model that replicates human selection. We use a deep neural network model trained in a real-world dataset of human motion to estimate which is the most probable behavior to be chosen by a human. Finally, the guidance trajectory is locally optimized with a Local Model Predictive Contouring Controller (LMPCC) (Brito et al. 2019) that ensures that the robot follows the desired social high-level behavior. The high-level behavior is dynamically updated to accommodate for unexpected changes in the predicted future trajectories of the surrounding pedestrians.

We extensively evaluate and analyze our proposed method in three different ways. First, we measure the accuracy of the supervised learning approach to predict the human behavior, and qualitatively examine some of the predictions. Second, we assess the behavior of our method with quantitative experiments in simulation and compare it with other state-of-the-art social motion planners. Finally, we conduct qualitative experiments on a real ground platform to evaluate and compare SHINE against state-of-the-art social motion planners in crowded and challenging real-world scenarios.

The article is organized as follows. The context and contribution of the work is studied in Section 2. Section 3 analyzes the problem to be solved, Section 4 presents and studies the prediction framework and Section 5 the navigation system. Section 6 and Section 7 examine the evaluation performed and, lastly, Section 8 concludes the work.

## 2 Related work and contribution

### 2.1 Social navigation

Navigating in environments populated by humans poses a challenge that has received significant attention from the robotics community, analyzed in recent survey papers as Mavrogiannis et al. (2023) or Singamaneni et al. (2024). Early obstacle avoidance approaches primarily focused on static obstacles, such as the dynamic window approach proposed in Fox et al. (1997), later improved by including moving obstacles in Missura and Bennewitz (2019). Velocity obstacles, introduced in Fiorini and Shiller (1998), laid the foundation for numerous studies, including Wilkie et al. (2009) or Lorente et al. (2018), which used the concept for modeling obstacle motion. Additionally, strategies like the Optimal Reciprocal Collision Avoidance (ORCA) (Van Den Berg et al. 2011) and other methods based on it like NH-ORCA (Alonso-Mora et al. 2012) or VR-ORCA (Guo et al. 2021) have explored the principle of reciprocity in collision avoidance. They consider a shared responsibility between

the robot and dynamic obstacles, facilitating collaborative maneuvers. In Lorente et al. (2018), an evolution of the velocity obstacles is presented, where strategies of passing before and after obstacles are applied to reach maximum speed and minimum time to reach the goal.

The motion planners in previously mentioned work use a local view of the environment in a short spatial horizon, which may lead to local traps and non-smooth trajectories. In Matínez-Baselga et al. (2023), the planner developed in Lorente et al. (2018) for dynamic environments is integrated with a global planner, avoiding trap situations. Some approaches use a Model Predictive Controller (MPC) to compute optimal control commands for the robot in advance. For example, LMPCC (Brito et al. 2019) is a method for real-time navigation among dynamic obstacles. Crowd motion estimation may be included in the optimizer to make it more reliable (Poddar et al. 2023).

The above approaches do not account for predictive capabilities of pedestrians, which can result in unsafe or unnatural behaviors (Kretzschmar et al. 2016). Some methods include social constraints or try to imitate human motion. One example is the early method Social Forces (Helbing and Molnar 1995) and its extended versions (Jiang et al. 2017), which intend to model social interactions as a sum of attractive and repulsive forces.

Previous methods typically treat other agents as dynamic obstacles whose trajectories are predicted, without fully addressing the complexities of interactions between decision-making agents. This may lead to defensive and opaque behaviors (Trautman et al. 2015). Thus, another important body of work addresses multi-agent interactions by coupling prediction and planning. Some methods do it explicitly using, for example, intent inference (Bandyopadhyay et al. 2013; Bai et al. 2015), a game-theoretic perspective (Fridovich-Keil et al. 2020; Sun et al. 2021; Peters et al. 2024; Mehr et al. 2023) or topological representations (Mavrogiannis et al. 2017; Mavrogiannis and Knepper 2019). Since each agent's action depends on the decisions of the others, explicitly solving the coupled prediction and planning problem presents a considerable computational challenge.

Other approaches implicitly accounts for interactions with crowds in their design. Particularly, deep reinforcement learning (DRL) planners show promising results by learning from interactions with the environment. A common approach is using decentralized end-to-end algorithms whose output is the velocity of the robot (Chen et al. 2017). Some of the networks use different layers to encode the environment and extract features to navigate, such as LSTM layers (Everett et al. 2018), (Everett et al. 2021), attention (Chen et al. 2019) or graph neural networks (Chen et al. 2020a), (Chen et al. 2020b). However, navigating with humans require special considerations, due to social norms and special needs in collaboration and safety. In Hu et al. (2022), social stress is introduced to the state space of the DRL framework, while in Zhou et al. (2022) and Martínez-Baselga et al. (2023), social attention is used to model the interactions between the humans and the robot. Nevertheless, DRL approaches have the limitations of being very dependent on the simulator and the modeling of people and robot interactions, as well as a lack of explicit collision constraints.

## 2.2 Topology-based navigation

Reactive planners focus on the immediate surroundings of the robot, while MPC-based planners optimize their current trajectory, leading to suboptimal paths. Two trajectories that start and end at the same points but cannot be transformed into one another without crossing an obstacle adopt different spatial configurations to avoid the obstacle. We say that they are topological distinct trajectories. Homotopy classes refer to different topological paths or trajectories that are considered equivalent in terms of their spatial relationship (further explained in Section 4). By computing trajectories in distinct homotopy classes, robots can choose trajectories that avoid obstacles and optimize their movements when unexpected obstacles or changes in the environment occur (Kolur et al. 2019). Topological information can be useful not only for navigation, but also for other tasks as information gathering (McCammon and Hollinger 2021). Discovering topologically distinct trajectories can be done with Voronoi diagrams (Rösmann et al. 2017a), but a more common approach in the literature are Probabilistic Roadmaps (PRM) (Siméon et al. 2000) or derivatives (Novosad et al. 2023), which is a sampling-based method to construct a sparse roadmap.

Some works use the idea of choosing one of the topologically distinct trajectories as a guidance of an MPC framework, producing a locally optimal trajectory that is topologically equivalent to the guidance. This has enabled fast and safe flights to UAVs with perception-aware trajectory replanning (Zhou et al. 2021), and demonstrated state-of-the-art performance for flights in cluttered environments (Penicka et al. 2022), (Penicka and Scaramuzza 2022).

More related to our problem, de Groot et al. (2023) presents an approach to navigate in dynamic environments by finding topologically distinct guidance trajectories in the dynamic space and use LMPCC to find a global optimal plan. Another interesting approach (Mavrogiannis et al. 2022) adds a cost in the MPC to encourage the robot to be topologically invariant in time. A recent approach (Ding et al. 2023) includes a homotopic cost in the generation of nodes of an Optimal Rapidly-exploring Random Trees (RRT\*) algorithm. Nevertheless, RRT\* is a planner for static environments and computes a global path regardless the future of the environment. Similar to our idea, H-TEB (Rösmann et al. 2017b) selects a topology class based on a cost learned by human parameters and plans a trajectory in that class. However, they only consider a static environment, use a cost function with only three parameters, need to predict the classes for the full scenario and assume the other agents have a collaborative behavior. Therefore, their navigation performance is strongly related to the collaboration of other agents and their predictions, which could be inaccurate due to the static selection of a topology in a dynamic environment.

A comparison of the main features of our proposed method with the discussed motion planners in dynamic environments is shown in Table 1.

## 2.3 Human navigation prediction

Understanding and predicting humans' motion is crucial for intelligent robots to navigate safely and interact in crowds, and there is a growing trend in research focused on predicting human trajectories (Rudenko et al. 2020). That knowledge could be used by the robot to use the same social navigation rules, ensuring safety, facilitating collaboration, adapting to social context and exhibiting natural human-robot interaction. Strategies include using social attention (Vemula et al. 2018), human-oriented transformers (Yuan et al. 2021) or solutions based on retrospective memory (Xu et al. 2022). In Salzmann et al. (2020), the authors present a modular model that incorporates agents dynamics and heterogeneous data, which they extend in Ivanovic et al. (2023) with adaptive meta-learning, showing that the method is easily extensible. These methods prove their performance in public real-world datasets, such as the ETH/UCY dataset (Lerner et al. 2007) (Pellegrini et al. 2009), which have data of pedestrians positions in 5 different scenarios.

A recent work, VOMP-h (Wakulicz et al. 2023), predicts homotopy classes. It discusses the problems of trajectory prediction based on geometric representations, explaining the benefits of predicting homotopy classes instead of the trajectories. However, this approach is not directly applicable to navigation and assumes a static environment. Furthermore, it estimates the partial homotopy class of the next obstacle faced by the robot, so it is hard to assess the scalability in dynamic and cluttered scenarios. Previously cited H-TEB (Rösmann et al. 2017b) also predicts the homotopy class followed by the humans. However, it does it in the scenarios as if they were static, not taking into account the past or the future of the environment, and uses a sum of three hand-crafted terms to predict the cost. The weight of the terms is learned from demonstrations.

Other work that predicts the topology of the environment is SCN (Mavrogiannis et al. 2017), where a network is trained to predict the permutations in the positions of people moving in a Social Force simulator, and includes a planner based on braids to select a braid that meets the predicted permutation. In Table 2, we summarize the main differences between our approach and these two methods. While SCN only learns permutations, VOMP-h only predicts homotopy classes in static environments and is not translated to navigation commands.

## 2.4 Contribution

The two-key contributions of this work are:

1. A new supervised learning approach, able to learn human navigation preferences in crowded environments. Our approach learns how different the topology class of a queried trajectory is from that which a human would choose. This allows us to distinguish between social and non-social high-level navigation decisions.
2. A new planner that infers a social high-level navigation decision to initialize an optimization-based planner (LMPCC (Brito et al. 2019)), and may reactively replan with new social decisions if the scenario behaves unexpectedly.

**Table 1.** A comparison of our navigation planner with other planners for dynamic environments. The social behavior column refers to social features included in the planner, social learning whether those features are learned, C.D.P. scenarios if the planner is tested in Crowded, Dynamic and Possibly non-cooperative scenarios, and global planning checks the ability of the planner to avoid local traps or replan a different path from the one previously followed.

Planner	Social behavior	Social learning	C.D.P scenarios	Global planning	Open source
LMPCC (Brito et al. 2019)			✓		✓
Guidance-MPCC (de Groot et al. 2023)			✓	✓	✓
Social Force (Helbing and Molnar 1995)	✓		✓		✓
Social DRL (Martinez-Baselga et al. 2023)	✓	✓	✓		✓
T-MPC (Mavrogiannis et al. 2022)	✓		✓		✓
SCN (Mavrogiannis et al. 2017)	✓	✓			
PRTIRL (Ding et al. 2023)	✓	✓		✓	
H-TEB (Rösmann et al. 2017b)	✓	✓		✓	
SHINE	✓	✓	✓	✓	✓

**Table 2.** A comparison of our learning path topologies approach with others. The dynamic environments' column refers to including dynamic obstacles in the predictions, navigation if the predictions may be applied in navigation, real-world data if it was used for training, homotopy class if they predict homotopy classes, and sophisticated model if the prediction model to capture humans' preferences is more complex than a simple addition of a few terms.

Algorithm	Dynamic environments	Navigation	Real-world data	Homotopy class	Sophisticated model	Open source
SCN (Mavrogiannis et al. 2017)	✓	✓			✓	
H-TEB (Rösmann et al. 2017b)		✓	✓	✓		
VOMP-h (Wakulicz et al. 2023)			✓	✓	✓	
SHINE	✓	✓	✓	✓	✓	✓

We evaluate how well our model predicts the human high-level navigation decisions in real-world scenarios and compare it with other variations and hand-crafted cost selections, and we show that our approach scales well in different scenarios and environments. Then, we extensively compare our approach with other motion planners in dynamic environments in both simulation and the real world, showing their differences and limitations. Additionally, we validate the behavior of our proposed planner in specific challenging scenarios and among five pedestrians in a tight space in a laboratory experiment, showing learned socially acceptable behaviors.

The system will be released as open source upon acceptance.

### 3 Problem formulation

We consider scenarios in which a robot must navigate to a goal position  $\mathbf{p}_{goal} = (x_{goal}, y_{goal})$  while ensuring collision avoidance with dynamic obstacles, i.e. humans. We model the robot's motion by the deterministic discrete-time non-linear dynamics:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad (1)$$

where  $\mathbf{x}_k \in \mathbb{R}^{n_x}$  is the state and  $\mathbf{u}_k \in \mathbb{R}^{n_u}$  the control input of the system at time step  $k$ , and  $n_x$  and  $n_u$  are the state and input dimensions, respectively. The state of the robot at time  $k$  contains its 2-D position  $\mathbf{p}_k = (x_k, y_k) \in \mathbb{R}^2$ .

We will use the superscript  $T$  when referring to the collection of variables over  $T$  discrete time steps. For example,  $\mathbf{p}_k^T = [\mathbf{p}_{k-T}, \dots, \mathbf{p}_k]$  denotes the robot's positions over the past  $T$  discrete time steps.

The robot and the dynamic obstacles are represented as disks with radius  $r_{robot}$  and  $r_{obs}$ , respectively. The position

of the obstacle  $j$  at time  $k$  is denoted as  $\mathbf{o}_{j,k} = (x_{j,k}, y_{j,k}) \in \mathbb{R}^2$  and the position of the set of  $M$  obstacles at time  $k$  is denoted as  $\mathcal{O}_k = [\mathbf{o}_{0,k}, \dots, \mathbf{o}_{M,k}]$ . We assume that the previous positions of the obstacles are known for a time horizon  $T$ , denoted as  $\mathcal{O}_k^T = [\mathcal{O}_{k-T}, \dots, \mathcal{O}_k]$ . The values of  $M$  and  $T$  are not fixed.

The state space that we consider for planning is composed by the workspace and time:  $\mathcal{X} = \mathbb{R}^2 \times [0, T]$ . A trajectory is a continuous path through the state space:  $\tau : [0, 1] \rightarrow \mathcal{X}$ , and  $\mathcal{X}_{traj}$  the set of all possible trajectories in  $\mathcal{X}$ . We denote  $\tau_1 \sqcup \tau_2$  the the curve formed by  $\tau_1$  and  $\tau_2$  together, and  $-\tau$  the same trajectory  $\tau$  but in opposite orientation. The collision-free space describes the state space without the space occupied by the obstacles.

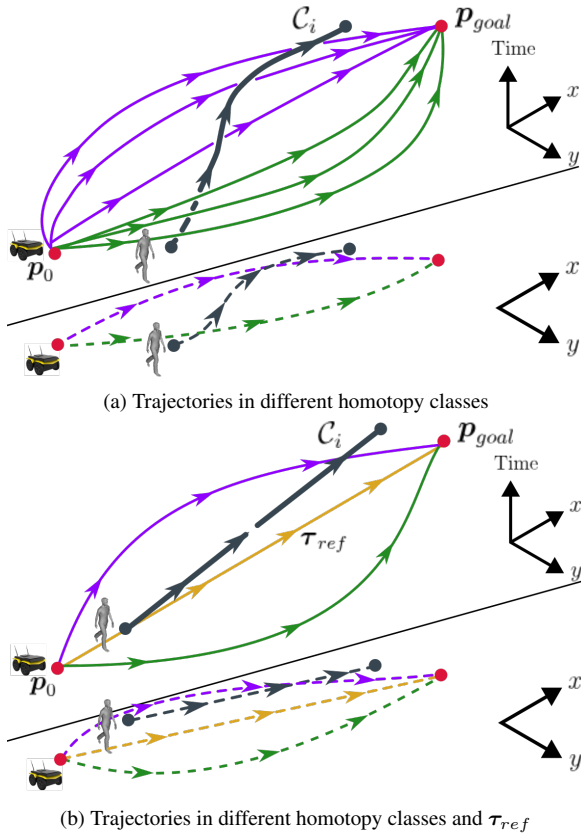
#### 3.1 Homotopy and Homology Classes

We are interested in the high-level behavior of trajectories in the collision-free space (i.e., how they pass the obstacles) so that we can learn this behavior from humans. The direction in which a trajectory passes obstacles is captured by *homotopy* classes, a concept from topology.

**Definition 1.** (Bhattacharya et al. 2012) (*Homotopic Trajectories*) Two trajectories,  $\tau_1 : [0, 1] \rightarrow \mathcal{X}$  and  $\tau_2 : [0, 1] \rightarrow \mathcal{X}$ , connected by the same starting and ending point, are homotopic if they can be continuously deformed into each other without intersecting with any obstacle, while keeping the endpoints. Homotopic trajectories belong to the same homotopy class.

An example of trajectories in two different homotopy classes is depicted in Fig. 2(a). In this work, we use the almost equivalent concept of *homology* classes, similar to





**Figure 2.** Schematic representation of trajectories starting at point  $p_0$  and ending at  $p_{goal}$ . Obstacles trajectory is  $C_i$ . Green and violet trajectories have the same homotopy class among them. 3-D (top) with one axis representing time and 2-D (bottom) representations are plotted in both (a) and (b). In (b) we introduce a reference trajectory  $\tau_{ref}$  (yellow) to assess the homotopy class of a single trajectory.

previous works (Bhattacharya et al. 2012; Rösman et al. 2017a).

**Definition 2.** (Bhattacharya et al. 2012) (*Homologous Trajectories*) Two trajectories,  $\tau_1 : [0, 1] \rightarrow \mathcal{X}$  and  $\tau_2 : [0, 1] \rightarrow \mathcal{X}$ , connected by the same starting and ending point, are homologous if the loop formed by them ( $\tau_1 \sqcup -\tau_2$ ) forms the boundary of a two-dimensional manifold that is immersed in  $\mathcal{X}$  and does not contain nor intersect with any obstacle. Homologous trajectories belong to the same homotopy class.

Two homotopic trajectories are always homologous. Even though two homologous trajectories may not be homotopic, they are very similar in most of robotic applications. Particularly, in the state space  $\mathcal{X}$ , two homologous trajectories are always homotopic (Bhattacharya et al. 2011). In this work, we use the homology classes concept because, contrary to homotopy classes, which are hard to compute, homology classes are comparable in practice through their H-signature (Bhattacharya et al. 2012). In the 3-dimensional state space, the H-signature can be computed by assuming that the time dimension progresses linearly (e.g., discrete-time). To do so, each obstacle  $i$  may be defined as a vector field that exhibits rotational symmetry around the time-axis, centered in the 3-dimensional obstacle trajectory  $C_i$ , as seen in Fig. 2(a). The vector field corresponding to obstacle  $i$ ,  $F_i$ ,

is constructed as an integral over  $C_i$ :

$$F_i(r) = \int_{C_i} \frac{dl \times (r - l)}{|r - l|^3} \quad (2)$$

where  $dl$  is an infinitesimal vector element along  $C_i$ ,  $l$  a point in  $C_i$  and  $r$  the point in which the field is being evaluated. This definition is analogous to the Biot-Savart law of electromagnetism.

An important property of  $F_i$  is that it is not path-independent, i.e., the integral of  $F_i$  between two points depends on the specific path taken, not only on the end-points. Nevertheless, the integral over  $F_i$  of trajectories that belong to the same homology class is the same. In Fig. 2(a), all violet trajectories and all green trajectories exhibit this property, respectively. Thus, path independence holds within homology classes. This is used to define signatures that represent homology classes, named H-signatures. For every trajectory  $\tau$ , having  $M$  obstacles in the environment, its H-signature  $\mathcal{H}_3 : \mathcal{X}_{traj} \rightarrow \mathbb{R}^M$  is a vector in  $\mathbb{R}^M$ :

$$\mathcal{H}_3(\tau) = [h_1(\tau), h_2(\tau), \dots, h_M(\tau)]^T, \quad (3)$$

where  $h_i(\tau)$  is defined by the integral:

$$h_i(\tau) = \int_{\tau} F_i(r) dr. \quad (4)$$

## 4 Homology class prediction

In this section, we formally define our version of H-signature and present our approach to predict the homology class selected by humans in crowded scenarios.

### 4.1 Modified H-signature

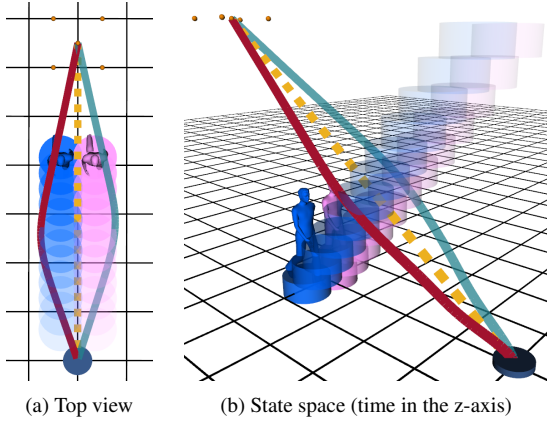
Equation 4 holds an important property. The integral of  $F_i$  over a closed loop  $\gamma$ ,  $\oint_{\gamma} F_i(r) dr$ , corresponds to the topological winding number of the loop around  $C_i$ , i.e., how many times  $\gamma$  loops around  $C_i$ . For example, if  $\oint_{\gamma} F_i(r) dr = 0$ , then  $\gamma$  does not enclose the obstacle; if  $\oint_{\gamma} F_i(r) dr = \pm 1$ , it encloses it, with the sign depending on the loop direction; and if  $\oint_{\gamma} F_i(r) dr = \pm n_w$ , it loops around it  $n_w$  times. This property may be used to compare the homology classes of two trajectories,  $\tau_1$  and  $\tau_2$ , connected to the same starting and ending points, by using the loop formed by  $\tau_1 \sqcup -\tau_2$ :

$$h_i(\tau_1 \sqcup -\tau_2) = \oint_{\tau_1 \sqcup -\tau_2} F_i(r) dr \quad (5)$$

Therefore,  $h_i(\tau_1 \sqcup -\tau_2)$  will be 0 if the trajectories do not enclose the obstacle  $i$ , and  $\pm 1$  if they do.

We define our own version of the H-signature using a reference trajectory  $\tau_{ref}$ , represented in Fig. 2(b) in yellow, and Equation 5. We use a reference trajectory to be able to assess the topology of a trajectory in isolation, consistently and invariantly. Having a trajectory  $\tau$  with initial and final times  $t_0$  and  $t_f$ , whose origin is  $\tau^0 = (x_o, y_o, t_0)$  and final point is  $\tau^f = (x_f, y_f, t_f)$ ; the reference trajectory is defined by the straight line that connects the points  $\tau^0$  and  $\tau^f$ .

Among the infinite set of trajectories, that could be chosen, we selected  $\tau_{ref}$  as the reference path for the signature as it is



**Figure 3.** An example of two trajectories, left-passing (red line) and right-passing (blue line), in distinct homology classes. The trajectories of the robot (blue disk) pass the future motion prediction of the humans (transparent cylinders) differently. The reference trajectory  $\tau_{ref}$  (dotted yellow) would lead to collision. Orange dots are other possible goal grid points computed by enlarging the goal with the robot radius.

the simplest trajectory that drives the robot to the goal if there were not obstacles (a straight line with constant velocity), and it forms a loop with  $\tau$ . We define the new signature for  $M$  obstacles as:

$$\mathcal{H}_{ref}(\tau) = [h_1(\tau_{ref} \sqcup -\tau), \dots, h_M(\tau_{ref} \sqcup -\tau)]^T \quad (6)$$

In practice, and due to our application of  $\mathcal{H}_{ref}$ , we use the absolute value of the integral as the H-signature, as the sign provides redundant information. In this way, for each obstacle  $i$ , the element  $i$  of  $\mathcal{H}_{ref}(\tau)$  will have the value 0 if  $\tau$  has the same topological behavior as  $\tau_{ref}$  with respect to  $i$  and  $n_w$  otherwise (usually 1 in our application, as robot trajectories are normally not supposed to loop around pedestrians). Unlike  $\mathcal{H}_3$  defined in Equation 3, the meaning of  $\mathcal{H}_{ref}$  may be easily understood by humans. Intuitively,  $\mathcal{H}_3$  encodes how much a trajectory loops around each of the obstacles, with positive or negative values depending on the looping side. Thus, the value of  $\mathcal{H}_3(\tau)$  depends on the starting and final position of  $\tau$  and the specific trajectories of the obstacles, whereas the value of  $\mathcal{H}_{ref}$  may be intuitively inferred. For example,  $\mathcal{H}_{ref}$  of the violet trajectory in Fig. 2(b) is [1] and  $\mathcal{H}_{ref}$  of the green one is [0]. Therefore, the benefits of the signature are that it is humanly understandable and invariant of the shape and direction of the trajectories, as well as to the scenarios and environments.

Every homology class may not have non-colliding and feasible trajectories when considering robot and human radius, specially in crowded environments. For example, Fig. 3 shows a scenario with two humans where the robot may only choose between trajectories in two homology classes. Even though  $\tau_{ref}$  leads to a collision,  $\mathcal{H}_{ref}$  may still be computed.

## 4.2 Homology class selection

While humans are constantly choosing a suitable topological path when navigating, manually defining a function that considers all factors that are involved in the decision-making process is impossible. We aim to learn the human selection

process so that, having a recorded real-world scenario where a human has followed the ground truth trajectory  $\tau^h$ , our method is able to predict  $\mathcal{H}_{ref}(\tau^h)$  without knowing  $\tau^h$ . This knowledge could be applied to plan a global guidance trajectory  $\tau^g$  that belongs to the same homology class as  $\tau^h$  and use a local planner to navigate within that class to achieve socially compliant navigation. However, sampling a guidance trajectory having a H-signature is a complex task that is sometimes unsolvable, since there may be H-signatures where all trajectories lead to collisions. Therefore, instead of simply trying to predict  $\mathcal{H}_{ref}(\tau^h)$  in a scenario, we propose sampling a guidance trajectory  $\tau^g$  and estimating the difference between  $\mathcal{H}_{ref}(\tau^g)$  and  $\mathcal{H}_{ref}(\tau^h)$ . The process is repeated for guidance trajectories that belong to every possible homology class in the environment, so that the one with the lowest difference is estimated to be the most social one.

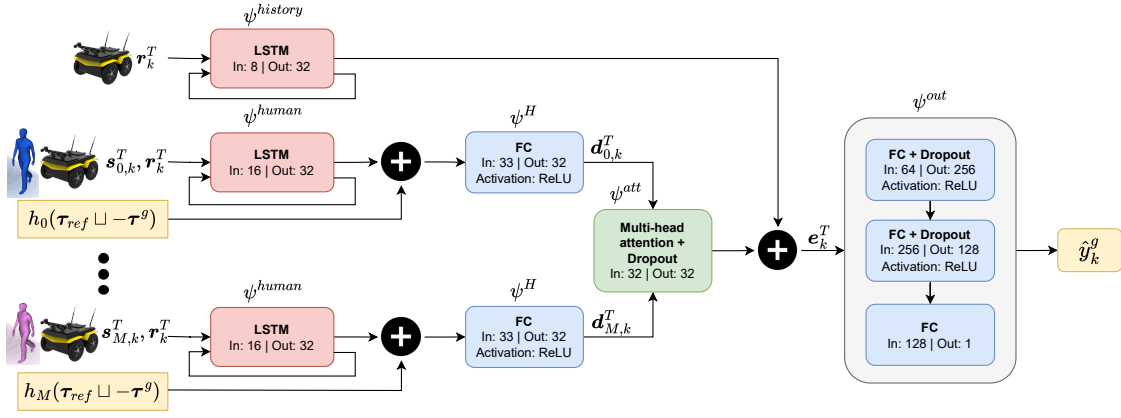
We consider a parametric cost function that estimates that difference using as inputs previous observations of the robot's positions  $p_k^T$  and the other humans' positions in the scenario  $\mathcal{O}_k^T$  over a time horizon  $T$  at time  $k$ . The cost function designed  $\mathcal{J}_\theta(\mathcal{H}_{ref}(\tau^g), p_k^T, \mathcal{O}_k^T)$  is approximated using a deep neural network with parameters  $\theta$ , represented in a diagram in Fig. 4. The goal of the network is to estimate the difference in H-signature (in a Euclidean sense) between the trajectory  $\tau^g$  and the trajectory  $\tau^h$  that a human would choose:

$$y^g = MSE(\mathcal{H}_{ref}(\tau^g), \mathcal{H}_{ref}(\tau^h)), \quad (7)$$

where  $y^g$  is the desired output of the network.

The network structure follows the architecture proposed in Salzmann et al. (2020) and Ivanovic et al. (2023). It shares the same encoding architecture with modifications to include the homology information, and fully connected layers to finally output the desired cost. The states of the robot and the surrounding humans are first extended by linearly interpolating the velocity  $(v_{x,k}, v_{y,k})$ , acceleration  $(a_{x,k}, a_{y,k})$ , and the heading angle  $\alpha$  from current and previous positions. Then, the considered full state of the robot at time  $k$  is  $r_k = [x_k, y_k, v_{x,k}, v_{y,k}, a_{x,k}, a_{y,k}, \sin \alpha_k, \cos \alpha_k]$  and the state of the human  $j$  is  $s_{j,k} = [x_{j,k}, y_{j,k}, v_{j,x,k}, v_{j,y,k}, a_{j,x,k}, a_{j,y,k}, \sin \alpha_{j,k}, \cos \alpha_{j,k}]$ . The full state of the robot and the human  $j$  over the last time horizon  $T$  are denoted as  $r_k^T = [r_{k-T}, \dots, r_k]$  and  $s_{j,k}^T = [s_{j,k-T}, \dots, s_{j,k}]$ , respectively, and the full state of all the obstacles in the environment is denoted as  $\mathcal{S}_k^T = [s_{0,k}^T, \dots, s_{M,k}^T]$ , given that there are  $M$  humans influencing the robot at time  $k$ .

The state of the robot over the time horizon,  $r_k^T$ , is encoded using an LSTM layer,  $\psi^{history}(r_k^T)$ , so that the previous states are considered while keeping time dependencies. The same approach is taken to encode the surrounding humans' state but, in this case, the state of each of the human  $j$  at each time step  $k$ ,  $s_{j,k}$ , is concatenated to the state of the robot at the same time step,  $r_k$ , and given as input to an LSTM encoder, using the same weights for every human:  $\psi^{human}(s_{j,k}^T, r_k^T)$ . The output of human  $j$  is concatenated to its corresponding value of  $\mathcal{H}_{ref}$  signature,  $h_j(\tau_{ref} \sqcup -\tau^g)$ . This is fed into a Fully Connected Layer (FCL),  $\psi^H$ , to get



**Figure 4.** Diagram with the layers of the network that we use to infer social high-level decisions. The inputs are the historical state of the robot,  $r_k^T$ , and the historical state of the  $M$  surrounding humans,  $s_{j,k}^T$ , at time  $k$  over the last time horizon  $T$  and the H-signature of the proposed guidance,  $\mathcal{H}_{ref}(\tau^g)$ . The output,  $\hat{y}_k^g$ , is the estimation of the result of Equation 7, which is the estimated difference of the homology class of guidance and the preference of a human.

an embedding vector,  $d_{j,k}^T$ , that encodes the information of the human  $j$  and its relation with  $\tau^g$ :

$$d_{j,k}^T = \psi^H(\psi^{human}(s_{j,k}^T, r_k^T), h_j(\tau_{ref} \sqcup -\tau^g)) \quad (8)$$

The output of the  $\psi^H$  layers is aggregated using attention ( $\psi^{att}$ ), as in Vemula et al. (2018), to combine all the influence that the neighbor obstacles have in the robot into a single vector, which is concatenated to the robot encoded state to get the final representation state,  $e_k^T$ :

$$e_k^T = \{\psi^{history}(r_k^T), \psi^{att}(d_{0,k}^T, \dots, d_{M,k}^T)\} \quad (9)$$

Finally,  $e_k^T$  is fed into a model with 3 fully connected layers,  $\psi^{out}$ . The two intermediate models, as well as  $\psi^H$ , include ReLU activation functions and dropout layers. The output is a single value, which is the cost of choosing the homology class:

$$\mathcal{J}_\theta(\mathcal{H}_{ref}(\tau^g), p_k^T, \mathcal{O}_k^T) = \hat{y}_k^g = \psi^{out}(e_k^T) \quad (10)$$

### 4.3 Training on real-world data

Real-world datasets of humans navigating in crowds are used to train the network, particularly the UCY (Lerner et al. 2007) and ETH (Pellegrini et al. 2009) datasets. The training tries to extract the behavior of every of the humans of the dataset. Because the data is prerecorded, the history and future motion of the humans are known. Each training sample consists of a planning problem for one of the humans where the future is used as ground truth. The network parameters  $\theta$  are optimized to fit the data:

$$\min_{\theta} \sum_{k=1}^{N_s} \sum_{g=1}^{N_h} \mathcal{L}_{pred}(\hat{y}_k^g, y_k^g), \quad (11)$$

where  $\mathcal{L}_{pred}$  is chosen to be Mean Squared Error,  $N_s$  are the number of scenarios in the dataset,  $N_h$  the number of homology classes found in scenario  $k$ ,  $\tau^g$  a guidance trajectory of a homology class,  $\hat{y}_k^g$  the output of the network for the guidance  $\tau^g$  in scenario  $k$  and  $y_k^g$  the difference between the H-signature of the guidance  $\tau_g$  and the actual

### Algorithm 1: Training

**Input:** Dataset of pedestrians navigating in a crowd in time  $\mathcal{D}$

**Output:** Updated network weights  $\theta$

```

1 for training_iterations do
2    $p_k^T, \mathcal{O}_k^T, \tau_k^h, \mathcal{O}_k^{fut} \leftarrow \text{SampleBatch}()$ 
   /* Sample batch with size  $B_s$  of different
   scenarios with previous and future
   agent and obstacles positions over a
   time horizon.  $k \in (0, \dots, B_s)$  is one
   element of the batch. */
3    $\mathcal{T}_k^* \leftarrow \text{SampleGuidances}(\mathcal{O}_k^{fut}, p_k, p_{goal})$ 
   /* Sample up to  $N$  topologically distinct
   guidance trajectories for all the  $B_s$ 
   scenarios ( $B_s \times N$  guidances). */
4    $y_k \leftarrow \text{MSE}(\mathcal{H}_{ref}(\mathcal{T}_k^*), \mathcal{H}_{ref}(\tau_k^h))$ 
   /* Compute the actual homological
   difference between the human's
   trajectory and each of the guidances
   ( $B_s \times N$ ). */
5    $\hat{y}_k \leftarrow \mathcal{J}_\theta(\mathcal{H}_{ref}(\mathcal{T}_k^*), p_k^T, \mathcal{O}_k^T)$ 
   /* Estimate the cost using the network
   (Real batch size is  $B_s \times N$ ). */
6    $\theta \leftarrow \text{UpdateWeights}(\theta, \mathcal{L}_{pred}(\hat{y}_k, y_k))$ 
7 end

```

trajectory followed by the human, as in Equation 7. The high-level training algorithm is represented in Algorithm 1.

The network is designed so that the different homology classes are loaded as different entries in the same batch. Therefore, the inference and training is done in parallel for all the homology classes, and there is no extra computational cost of having more or less homology classes.

**Accounting for scalability.** Our approach exhibits effective scaling in three different ways. First, it may have inputs with different numbers of previous timesteps, such as humans that were not previously sensed (their trajectory is shorter). This is achieved by using LSTM layers to process the temporal data used for the input of the network. Second, the number of surrounding humans is not fixed

and does not need to be always the same, as the attention layer deals with it. Finally, it is scalable in the lengths and shapes of the trajectories, as the H-signature depends on the obstacles instead of the specific guidance trajectory. We show this in Section 6.2 by testing the method with a prediction horizon much higher than the one used in training. In addition, the network could be easily extended to include other features such as the map of the environment as in the Trajectron++ (Salzmann et al. 2020) original work.

**Accounting for multi-modality.** Human navigation is multi-modal. In our approach, each of the navigation modes are represented with homology classes. A cost is assigned to each of the homology classes and the one with the minimum cost is chosen, as it is the one estimated to be the most likely to be followed by a human. As the cost is computed for all the homologies, the ones with similar cost are similarly acceptable. Nonetheless, in our navigation approach, the robot motion should be predictable, so the most probable mode is always chosen (the one with the minimum cost).

## 5 Navigation system

In the following, we present a navigation framework which derives topologically distinct trajectories, selects one guidance trajectory using the learned cost function, and applies a local motion planner to track it. A schematic representation of the three main components of the system is provided in Fig. 5.

### 5.1 Guidance trajectories proposal

We use the algorithm proposed in de Groot et al. (2023) to compute a sparse representation of paths (guidance trajectories) that go from the initial position of the robot to the goal, and use the H-signature to filter the trajectories that belong to the same homology class.

The method is built on Visibility-PRM and works in the same state space introduced in the problem formulation, composed by the workspace and time  $\mathcal{X} = \mathbb{R}^2 \times [0, T]$ . The idea is to keep a graph of guard and connector nodes initialized with the initial and final position of the desired trajectory as guard nodes. New nodes are randomly sampled and added to the graph if they can be connected without colliding with static and dynamic obstacles. If a new node connects two guards, it is added as a connector. Otherwise, it is included as a guard. If two connectors connect the same two guards and the paths they construct are homologous, only the connector with the shortest path is kept. Therefore, the result of the algorithm is a graph with a set of trajectories  $\mathcal{T}^*$  where  $\mathcal{H}_{ref}(\tau_1) \neq \mathcal{H}_{ref}(\tau_2), \forall (\tau_1, \tau_2) \in \mathcal{T}^*$ . Each of the resulting trajectories is topologically distinct and represents a homology class that the robot may follow to reach the goal.

The minimum-cost guidance trajectory, having the cost of Equation 10, can be selected as:

$$\tau_k = \arg \min_{\tau_{i,k} \in \mathcal{T}_k^*} \mathcal{J}_\theta(\mathcal{H}_{ref}(\tau_{i,k}), \mathbf{p}_k^T, \mathcal{O}_k^T) \quad (12)$$

### 5.2 Local Optimization-based Planner

The selected guidance trajectory identifies the most suitable homology class but is not guaranteed to be dynamically

feasible or collision free. To obtain a high-quality trajectory, we initialize and track the guidance trajectory with a local optimization-based planner that refines the trajectory in the same state space, similar to (de Groot et al. 2023). The local planner is an MPC that solves a trajectory optimization problem, enforcing dynamic and collision avoidance constraints and returns a smooth trajectory near the guidance trajectory. We use the planner LMPCC (Brito et al. 2019). From the guidance trajectory, we first derive a guidance *path*,  $[0, S] \rightarrow \mathbb{R}^2$  that maps the distance along the path (of length  $S$ ) to an  $x, y$ -position. The MPC solves the optimization problem

$$\min_{\mathbf{u} \in \mathbb{R}^{n_u}, \mathbf{x} \in \mathbb{R}^{n_x}} \sum_{k=0}^N J_k \quad (13a)$$

$$\text{s.t.} \quad \mathbf{x}_0 = \mathbf{x}_{\text{init}} \quad (13b)$$

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, \dots, N-1 \quad (13c)$$

$$g(\mathbf{x}_k, \mathbf{o}_{j,k}) \leq 0, \quad k = 1, \dots, N. \quad (13d)$$

The cost function at step  $k$ , with weights  $w$ , is given by

$$J_k = w_c J_{c,k} + w_l J_{l,k} + w_v J_{v,k} + w_a J_{a,k} + w_\omega J_{\omega,k}, \quad (14)$$

where  $J_{c,k}$  and  $J_{l,k}$  are contour and lag costs that track the guidance path as defined in (Brito et al. 2019),  $J_{v,k} = \|v_k - v_k^{\text{guidance}}\|_2^2$  tracks the velocity along the guidance trajectory and,  $J_a = \|a\|_2^2$  and  $J_\omega = \|\omega\|_2^2$  weigh the control inputs. The constraints in Eqs. (13b) and (13c) constrain the initial state and dynamics, respectively. Collision avoidance is specified through the constraints  $g(\mathbf{x}_k, \mathbf{o}_{j,k}) \leq 0$ ,

$$g(\mathbf{x}_k, \mathbf{o}_{j,k}) = 1 - (\Delta \mathbf{p}_{j,k})^T \mathbf{R}(\phi)^T \begin{bmatrix} \frac{1}{r^2} & 0 \\ 0 & \frac{1}{r^2} \end{bmatrix} \mathbf{R}(\phi) (\Delta \mathbf{p}_{j,k}), \quad (15)$$

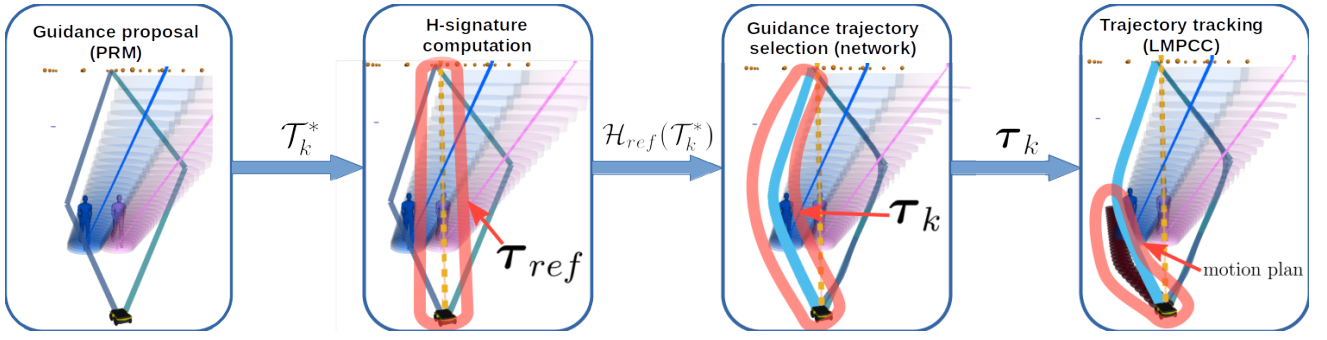
where  $\Delta \mathbf{p}_{j,k} = \mathbf{p}_k - \mathbf{o}_{j,k}$ ,  $\mathbf{R}(\phi)$  is a rotation matrix with orientation  $\phi$  of the robot and  $r = r_{\text{robot}} + r_{\text{obs}}$ . The constraints specify that the robot's and obstacles' discs should not overlap. The robot dynamics are modeled as a second-order unicycle model (Siegwart and Nourbakhsh 2011). We solve the online optimization with Forces Pro (Domahidi and Jerez 2014).

### 5.3 Enforcing consistent decisions

At every time step  $k$ , new guidance trajectories are sampled and a new homology class is selected. In this way, our approach may react to changes in environments with fast replanning capabilities, accounting for estimation errors or changes in the behavior of the pedestrians. Nevertheless, fast switching between trajectories in different topology classes can lead to indecisive and ultimately non-social navigation.

The problem is partially solved by the network, as people's intentions are usually constant in time and that behavior is intrinsically learned. In addition, we include a consistency weight  $w_c \in [0, 1]$  to encourage the robot to keep navigating within the same homology class. The weight is multiplied by the estimated network cost only if the homology class of a trajectory guidance is the same as the homology class followed in the previous control period, reducing its selection cost.





**Figure 5.** Schematic representation of SHINE. First, the guidance proposal algorithm in Section 5.1 finds trajectories (two blue lines originating from the robot) in different homology classes ( $\mathcal{T}_k^*$ ). Second, each trajectory is compared to a reference trajectory to identify its homology class. Third, the guidance selection component in Section 4, that was trained to select socially compliant trajectories from human motion data, selects the guidance trajectory ( $\tau_k$ ). Finally, the MPC-based planner in Section 5.2 computes the motion commands based on  $\tau_k$ .

The high-level navigation algorithm is represented in Algorithm 2, where the consistency weight is applied in line 6.

---

#### Algorithm 2: Navigation

---

**Input:** Previous robot positions  $\mathbf{p}_k^T$ , robot goal  $\mathbf{p}_{goal}$ , previous obstacle position  $\mathcal{O}_k^T$ , and predictions of future positions of the obstacles  $\mathcal{O}_k^{fut}$

**Data:** Consistency weight  $w_c$

```

1 for Every control period do
2    $\mathcal{T}_k^* \leftarrow \text{SampleGuidances}(\mathcal{O}_k^{fut}, \mathbf{p}_k, \mathbf{p}_{goal})$ 
3    $\hat{\mathbf{y}}_k \leftarrow \mathcal{J}_\theta(\mathcal{H}_{ref}(\mathcal{T}_k^*), \mathbf{p}_k^T, \mathcal{O}_k^T)$ 
   /* Estimate the cost of every guidance using the network. */
4   for  $\tau_{i,k} \in \mathcal{T}_k^*$  do
5     if  $\mathcal{H}_{ref}(\tau_{i,k}) = \mathcal{H}_{ref}(\tau_{k-1})$  then
6        $\hat{\mathbf{y}}_k^i \leftarrow \hat{\mathbf{y}}_k^i * w_c$ 
       /* Apply consistency weight. */
7     end
8   end
9    $\tau_k \leftarrow \text{argmin}(\mathcal{T}_k^*, \hat{\mathbf{y}}_k)$ 
   /* Select guidance with the minimum estimated cost. */
10   $\mathbf{u}_k \leftarrow \text{LMPCC}(\tau_k)$ 
   /* Use LMPCC to get the control commands. */
11   $\text{ApplyControl}(\mathbf{u}_k)$ 
12 end
```

---

## 6 Evaluation

This section discusses the experimental setup and results. We explain the experiments we conducted using real-world datasets in Section 6.2, in simulation in Section 6.3, and in a real platform in Section 6.4.

### 6.1 Experimental setup

**Training setup.** The LSTM and attention layer parameters are initialized with the pretrained weights of Trajec-tron++ (Ivanovic et al. 2023) provided in their original work, to leverage their feature extraction. Some of the most

**Table 3.** Training parameters used in the network.

Parameter	Value
Batch size	32
Training epochs	10
Learning rate	0.0001
Optimizer	Adam
History time horizon	2.8 s
Prediction time horizon	4.8 s
Dropout probability	0.1

important parameters of the network training are indicated in Table 3. We included L2 regularization. We used a history time horizon of 2.8 s and a prediction horizon of 4.8 s, as they are commonly employed in the ETH/UCY dataset.

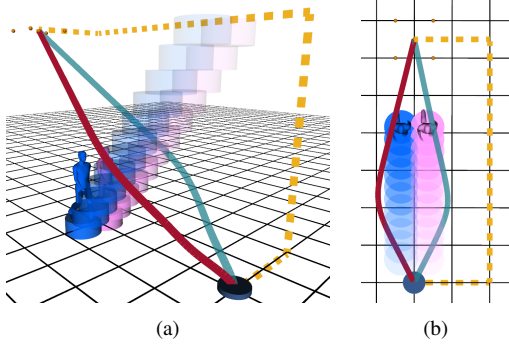
**Hardware setup.** The real-world evaluation is conducted on a Clearpath Jackal mobile robot in a room of size 7.5 m x 6.5 m. The platform is equipped with an Intel i5 CPU@2.6GHz. The robot and pedestrian positions are determined through a marker-based tracking system. Pedestrian predictions are based on the assumption of constant velocity, with the velocity obtained using a Kalman filter.

The motion planner is implemented in C++/ROS and the homology selector in Python. The whole system will be released as open source.

### 6.2 Prediction evaluation

The capacity of the network to select the correct guidance trajectory is evaluated in these experiments. To do so, we use testing scenarios of the ETH/UCY dataset where more than one homology class could be selected to navigate. We designed an accuracy score that measures the number of times the homology class actually chosen by the human is the one with the lowest cost. In the ETH/UCY dataset, there are five different environments: ETH, Hotel, University, Zara1 and Zara2. To evaluate the network and its generalization capabilities, it was trained in four scenarios and evaluated in the other one (Leave One Out).

As baselines, we propose selecting the guidance trajectory using three hand-crafted costs based on heuristics, taking  $N_\tau$  samples of positions  $\mathbf{p}_i$  and accelerations  $\mathbf{a}_i$  at constant time intervals along the trajectories:



**Figure 6.** Two views of  $\tau_{right}$  trajectory (yellow) and two trajectories that belong to different homology classes regarding two obstacles, where the axis-z represents time. In (a), the trajectories are represented with the time axis, while (b) is a top view of (a). The  $\mathcal{H}_{right}$  of the blue trajectory is  $[0,0]$  and the one of the red trajectory is  $[1,1]$ .

- Minimum length: The shortest guidance trajectory has the lowest cost:  $\mathcal{J}_{len} = \sum_{i \in N_\tau} \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$
- Minimum acceleration: The smoothest trajectory has the minimum cost:  $\mathcal{J}_{acc} = \sum_{i \in N_\tau} \alpha^i \|\mathbf{a}_i\|$ , where  $\alpha \approx 1$  to discount accelerations in time (0.95 in the implementation).
- Mixed cost: Combine previous costs:  $\mathcal{J}_{mix} = \mathcal{J}_{len} + \mathcal{J}_{acc}$

Additionally, we tried an alternative H-signature,  $\mathcal{H}_{right}$  to support the use of our selected H-signature,  $\mathcal{H}_{ref}$ . We define the new H-signature in the same way as in Section 4.1 with a reference trajectory that surrounds the whole environment through the right side,  $\tau_{right}$ . Having a trajectory  $\tau$  with initial and final times  $t_0$  and  $t_f$ , whose origin is  $(x_o, y_o, t_0)$  and final point is  $(x_f, y_f, t_f)$ ; and a set of positions of obstacles in that time interval  $\mathcal{O}_{t_f-t_0}^{t_f-t_0}$  where the set of coordinates in the y-axis of the all the positions are  $\mathbf{y}_O$ ; the reference trajectory is defined by the straight lines that connect the following points:

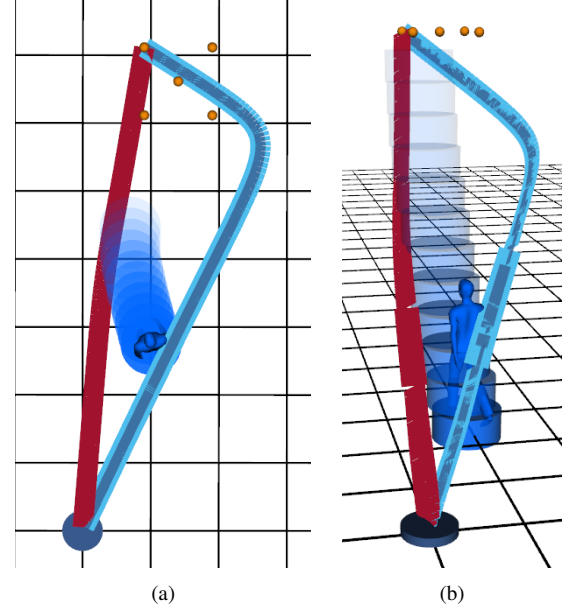
1.  $\tau_{right}^0 = (x_o, y_o, t_0)$
2.  $\tau_{right}^1 = (x_o, y_o, t'_0), t'_0 < t_0$
3.  $\tau_{right}^2 = (x_o, y_{right}, t'_0), y_{right} < y \ \forall y \in \mathbf{y}_O$
4.  $\tau_{right}^3 = (x_o, y_{right}, t'_f), t'_f > t_f$
5.  $\tau_{right}^4 = (x_f, y_{right}, t'_f)$
6.  $\tau_{right}^5 = (x_f, y_f, t'_f)$
7.  $\tau_{right}^6 = (x_f, y_f, t_f)$

The trajectory is represented in Figure 6, in the same way  $\tau_{ref}$  was represented in Figure 3. With  $M$  humans influencing the robot, we define the new signature as:

$$\mathcal{H}_{right}(\tau) = [h_1(\tau_{right} \sqcup -\tau), \dots, h_M(\tau_{right} \sqcup -\tau)]^T, \quad (16)$$

For every obstacle  $i$ ,  $\mathcal{H}_{right}(\tau)$  will be 0 if  $\tau$  avoids obstacle  $i$  through the right side, and 1 if it avoids it through the left. The network was trained using  $\mathcal{H}_{right}$  instead of  $\mathcal{H}_{ref}$  to test this other signature. In this section, we refer as  $\mathcal{J}_{\theta, ref}$  the cost estimated by the network using  $\mathcal{H}_{ref}$ , defined in Equation 10; and  $\mathcal{J}_{\theta, right}$  the alternative using  $\mathcal{H}_{right}$ .

The accuracy results are shown in Table 4. Our method significantly outperforms the other approaches. We noticed



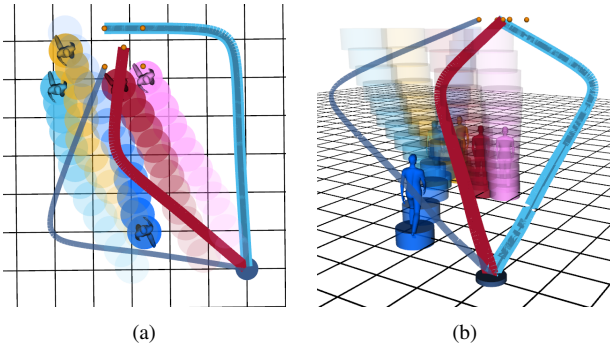
**Figure 7.** Top (a) and side (b) view of a scenario of the real-world dataset where the guidance chosen by our method (light blue) avoids passing through the pedestrian trajectory and is the one chosen by the real human. The guidance trajectory chosen with  $\mathcal{J}_{len}$  is represented in red.

that, once the human makes a decision on how to pass other humans, it usually sticks with it, following a previously made decision. This explains why all methods achieve good accuracy ( $> 0.5$ ). The results thus show that our method does not only select just the homology class the human was previously following, but also accurately estimates the decision at any time step.

**Table 4.** Accuracy metrics of the predictions of the human homology class.

Cost	ETH	Hotel	Univ	Zara1	Zara2	Avg.
$\mathcal{J}_{len}$	0.770	0.808	0.730	0.786	0.767	0.772
$\mathcal{J}_{acc}$	0.623	0.617	0.625	0.516	0.645	0.605
$\mathcal{J}_{mix}$	0.730	0.797	0.729	0.779	0.758	0.759
$\mathcal{J}_{\theta, right}$ (ours)	0.800	0.729	0.686	0.851	0.678	0.749
$\mathcal{J}_{\theta, ref}$ (ours)	<b>0.984</b>	<b>0.951</b>	<b>0.942</b>	<b>0.955</b>	<b>0.925</b>	<b>0.951</b>

For example, Figure 7 and Figure 8 show scenarios of the dataset where the guidance chosen with  $\mathcal{J}_{len}$  cost (red) is different from the one chosen with the network (light blue), and the one chosen with the network corresponds to the actual recorded behavior of the human. In both figures, the future positions of the humans are represented with circles increasing linearly in the z-axis, which represent the time. The guidance trajectories are represented as lines from the initial position to the goal, represented with the area with small orange spheres. In Figure 7, the optimal behavior in terms of path length is passing through the left of the blue pedestrian. Nevertheless, this could be socially rude, as it is overtaking the pedestrian from behind while intersecting with its current trajectory, so the network proposes avoiding it through its right side. In Figure 8, the guidance chosen is avoiding every pedestrian through the right, as it is the less invasive option from the three proposed guidance trajectories.



**Figure 8.** Top (a) and side (b) view of a scenario of the real-world dataset where the guidance chosen by our method (light blue) is less intrusive and is the one chosen by the real human. The guidance trajectory chosen with  $\mathcal{J}_{len}$  is represented in red, while in dark blue another guidance trajectory is represented.

In addition, it is interesting to see that in Table 4 the metrics are consistent among the different unseen environments. This highlights the fact that designing a cost that adapts to different scenarios is hard, while our approach may learn how to navigate in different scenarios.

The other conclusion extracted from the table is the benefit of using our signature. The input information of the network using  $\mathcal{H}_{ref}$  and  $\mathcal{H}_{right}$  is the same but with different encoding, but the information of  $\mathcal{H}_{ref}$  is easier for the network to understand. The information encoded by  $\mathcal{H}_{ref}$  follows the human reasoning. We hypothesize that, when a human tries to reach a goal, it first thinks about the shortest path, and then focus on what obstacles to avoid with respect to that path.

**Scalability.** The network faces three different types of scalability in the predictions: In the number of homologies available in the scenario, the number of surrounding humans and the trajectory guidance length. The scalability among the number of homologies and the surrounding humans is implicitly tested in the previous experiment, as they are variable among different time steps among every environment. To test scalability over the trajectory length, we repeat the previous experiment with a higher dataset prediction time horizon to get farther navigation goals. The previous horizon was set to 4.8 s, as it is a common choice in most of the works, and we double it to 9.6 s for this experiment. Note that the network has the same fixed weights (with no fine-tuning or retraining) as before, trained with the short time horizon. The accuracy metrics are shown in Table 5. The accuracy of our method is still significantly higher than the other approaches, due to the fact that the network is not dependent on the trajectory guidance, but on the signature of the homology class, so the weights are not overfitting the trajectory length.

### 6.3 Simulation experiments

We conducted experiments in a battery of 50 simulation scenarios and gathered behavior metrics, similar to the ones proposed in Biswas et al. (2022) or Francis et al. (2023). The simulated environment (Figure 9) is a corridor with 12 pedestrians that pass through the corridor and avoid other pedestrians and the robot using Social Force (Helbing and

**Table 5.** Accuracy metrics of the predictions of the human homology class, taking data from the dataset with a prediction horizon of 9.6 s.

Algorithm	ETH	Hotel	Univ	Zara1	Zara2	Avg
$\mathcal{J}_{len}$	0.889	0.828	0.688	0.650	0.759	0.763
$\mathcal{J}_{acc}$	0.722	0.797	0.697	0.676	0.783	0.735
$\mathcal{J}_{mix}$	0.818	0.807	0.702	0.705	0.749	0.756
$\mathcal{J}_{\theta, right}$ (ours)	0.727	0.720	0.614	0.764	0.620	0.689
$\mathcal{J}_{\theta, ref}$ (ours)	<b>1.000</b>	<b>0.954</b>	<b>0.780</b>	<b>0.789</b>	<b>0.880</b>	<b>0.881</b>

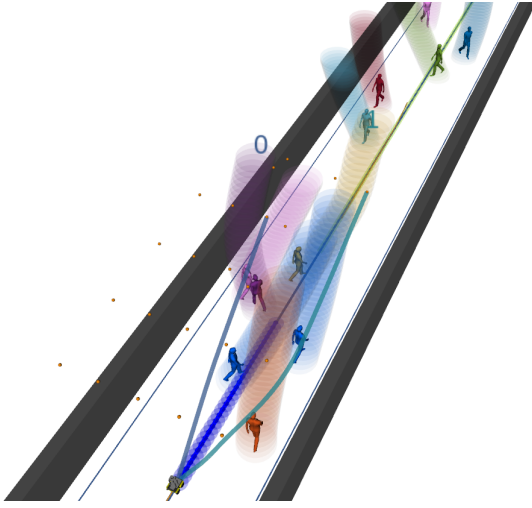
Molnar 1995). The simulated robot has differential-drive restrictions, and its task is to navigate through the corridor for 25 m. We gather metrics for a differential drive version of Social Force, a social DRL-based planner (Martinez-Baselga et al. 2023), an MPC-based planner (LMPCC) (Brito et al. 2019), and SHINE. As the metrics have different magnitudes, for a better understanding and comparison, we express the metrics related to Social Force, selected as reference. The results are represented in Table 6.

**Table 6.** Social metrics of the different planners. They are expressed in relation to Social Force metrics (except success and collision rates). The arrows indicate whether higher or lower values are preferable.

Metric	Social Force	DRL	LMPCC	SHINE
Success rate $\uparrow$	0.74	0.90	<b>0.98</b>	<b>0.98</b>
Collision rate $\downarrow$	0.26	0.10	<b>0.02</b>	<b>0.02</b>
Avg. path length $\downarrow$	<b>1.000</b>	1.014	1.010	1.009
Avg. time to goal $\downarrow$	1.000	<b>0.997</b>	1.052	1.003
Avg. speed $\uparrow$	1.000	1.005	0.968	<b>1.010</b>
Min. dist. obs. $\uparrow$	1.000	<b>1.043</b>	0.996	1.011
Avg. dist. obs. $\uparrow$	1.000	1.020	<b>1.030</b>	1.022
Min. time to coll. $\uparrow$	1.000	<b>1.091</b>	0.827	1.069
Path irregularity $\downarrow$	1.000	2.093	<b>0.793</b>	0.814
Avg. $\omega$ $\downarrow$	1.000	2.662	0.389	<b>0.311</b>
Avg. acceleration $\downarrow$	1.000	1.537	1.116	<b>0.805</b>
Avg. jerk $\downarrow$	1.000	1.673	1.080	<b>0.760</b>

In these scenarios, both LMPCC-based planners (LMPCC and SHINE) are safer with higher success rates (number of times the robot completes the task without collisions or getting stuck per scenario) and lower collision rates (number of times the robot collides per scenario). Social Force’s goal is navigating in a way that respects social norms with physics-inspired principles of attractive and repulsive forces, and its simplicity compared to the other planners could contribute to a higher collision rate. The DRL planner was trained in CrowdNav simulator, which uses random positions for humans. Its success rate could increase after training it in corridor scenarios, but it would mean that its effectiveness does not translate to different settings. The metrics of average path length, time to reach the goal, speed, distance to obstacles and time to collision are similar and comparable for all the planners; as their meaning is more related to the local constraints, and we set their parameters to behave similarly.

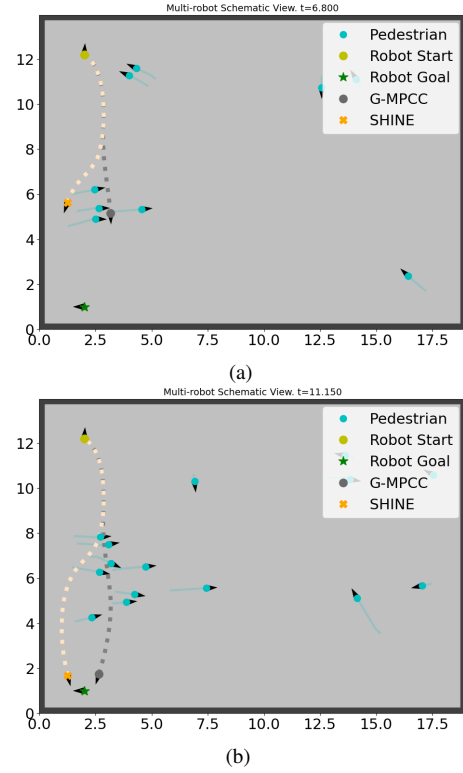
There is a clear difference in path irregularity (angle between the robot heading and the vector pointing to the goal) and the angular velocity ( $\omega$ ) between the MPC-based planners and the purely reactive ones. The reason why this happens is that these methods plan the whole path in advance, leading to smoother trajectories, which



**Figure 9.** Simulated corridor environment where different planners are tested. The predicted trajectories of the pedestrians are represented with semi transparent shadows of different colors, the plan with an intense blue and two guidance trajectories with two lines that start in the robot.

are preferable in social navigation. In addition, there is a difference in average acceleration and jerk (time derivative of acceleration) between our planner and the rest, as it is able to escape from local minima and replan when the environment changes, instead of having to suddenly accelerate or brake to avoid collisions.

Finally, we conducted experiments to assess the benefit of learning the guidance trajectories from humans rather than designing a hand-crafted cost for it. To do so, we used SocNavBench (Biswas et al. 2022), a simulator that generates humans using prerecorded trajectories, to compare SHINE with Guidance-MPCC using  $\mathcal{J}_{len}$ ,  $\mathcal{J}_{acc}$  and  $\mathcal{J}_{mix}$ . In this way, we intend to have more realistic simulations that resemble the real world and to use an existing benchmark. The metrics gathered for 5 runs of the 33 scenarios proposed as a benchmark in SocNavBench are shown in Table 7. We do not include comparisons with other planners of the simulator, as they do not consider robot dynamics. In this simulator, humans are spawned at the time when the recorded data of each specific human starts. Therefore, humans are sometimes spawned right into the robot position, resulting in an unavoidable collision and not reliable metrics. In addition, some scenarios may not have trajectories in different homology classes as either the robot faces no obstacles in its way to the goal or the scenario is too crowded. Therefore, differences in Table 7 among the methods are small. Thus, we selected the benchmark scenarios where SHINE and  $\mathcal{J}_{acc}$ , which has the best performance in Table 7 among the hand-crafted costs, navigated within different homology classes without collisions. The resulting metrics in those eleven scenarios are shown in Table 8. The results prove a better performance of SHINE with respect any other alternative used as baseline when it chooses a different homology class. One example is shown in Fig. 10, where both G-MPCC and SHINE reach the goal successfully at similar times, but SHINE in a way that is less intrusive regarding pedestrians. The metrics show that SHINE exhibits safety with high success rates, efficiency with low task



**Figure 10.** A scenario (named *univ\_trapTL*) of SocNavBench showing SHINE and G-MPCC with  $\mathcal{J}_{acc}$  cost trajectories.

duration and path length ratio metrics, as well as higher minimum distance to pedestrians and minimum time to collision metrics per episode. It is specially remarkable how it reaches the goal in considerably less time while keeping the highest time to collision to pedestrians.

**Table 7.** Metrics gathered in SocNavBench.

Metric	$\mathcal{J}_{len}$	$\mathcal{J}_{acc}$	$\mathcal{J}_{mix}$	SHINE
Success rate $\uparrow$	0.850	<b>0.912</b>	0.856	<b>0.912</b>
Duration (s) $\downarrow$	12.676 (2.7)	12.211 (2.5)	12.663 (2.7)	<b>11.994</b> (2.6)
Path length ratio $\downarrow$	0.976 (0.0)	0.964 (0.0)	0.976 (0.0)	<b>0.957</b> (0.0)
Min. closest ped. dist. (m) $\uparrow$	0.880 (0.9)	0.942 (0.8)	0.879 (0.8)	<b>0.945</b> (0.8)
Min. time to coll. $\uparrow$	1.065 (2.1)	1.114 (2.0)	1.039 (2.1)	<b>1.141</b> (2.0)

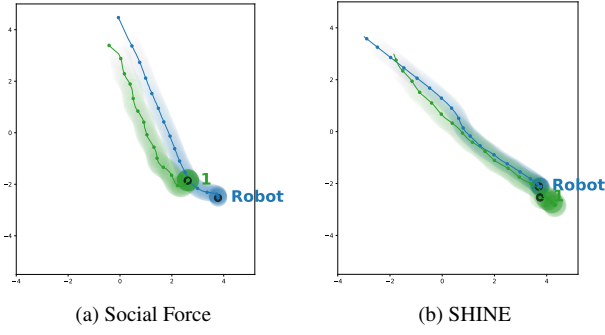
**Table 8.** Metrics gathered in SocNavBench in scenarios where SHINE and Guidance-MPCC with  $\mathcal{J}_{acc}$  are in different homology classes.

Metric	$\mathcal{J}_{len}$	$\mathcal{J}_{acc}$	$\mathcal{J}_{mix}$	SHINE
Success rate $\uparrow$	0.909	<b>1.000</b>	0.909	<b>1.000</b>
Duration (s) $\downarrow$	13.735 (3.4)	13.223 (2.1)	13.740 (2.7)	<b>12.527</b> (2.6)
Path length ratio $\downarrow$	0.993 (0.0)	0.992 (0.0)	1.016 (0.1)	<b>0.972</b> (0.0)
Min. closest ped. dist. (m) $\uparrow$	0.523 (0.0)	0.522 (0.0)	0.517 (0.0)	<b>0.523</b> (0.0)
Min. time to coll. $\uparrow$	0.401 (0.2)	0.368 (0.2)	0.360 (0.2)	<b>0.444</b> (0.3)

## 6.4 Real world experiments

We evaluated our approach in the real platform and compared it with other methods of the state of the art. We conducted two kind of experiments. First, a group of experiments to assess the limitations of other methods and the behavior of SHINE in those situations. We designed simple scenarios which could make planners fail due to their design. Particularly, we observed problems of Social Force in what we called push out scenarios, Social DRL in scenarios different from what experienced in training and LMPCC in zig-zag scenarios. The second type of experiments were





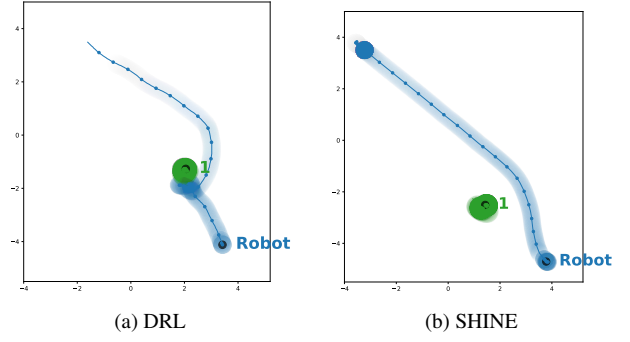
**Figure 11.** Graphs of the result of a push out scenario with Social Force (a) and SHINE (b) planners. The robot is represented in blue and humans in green. The starting positions are represented with a black circle and the trajectory in time with a shadow with increasing transparency.

conducted to test the navigation capabilities in crowded environments. In every experiment, the robot is placed in a corner of the room and has to navigate to the other corner. A visual summary of the real-world experiments is recorded in the supplementary video, which may be accessed in the project repository (Martinez-Baselga et al. 2024).

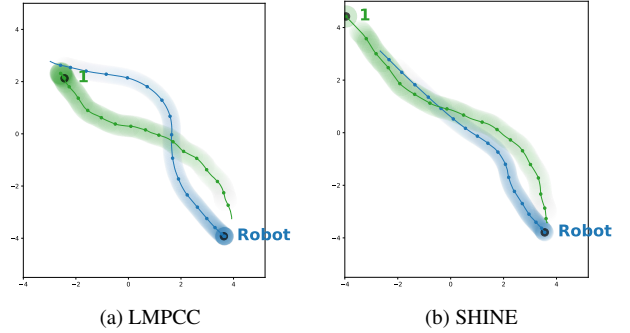
**6.4.1 Push out scenario** We tried an experiment where a human walks in the same direction the robot is trying to avoid it, closing its way. Social Force (Helbing and Molnar 1995) exhibits limitations in performance in scenarios where the attractive and repulsive forces are opposing, as in Figure 11 (a). Thus, the resulting behavior of this planner is that the robot keeps trying to avoid the human in that way and, therefore, the human pushes the robot out of the original way. In these situations, SHINE makes the robot brake or avoid the human through the other side, since it plans a long-term trajectory (Figure 11 (b)).

**6.4.2 Static scenario** End-to-end DRL methods reactively send the robot the velocity commands they estimate to provide the highest reward. Their limitations are that they do not provide guarantees for collision avoidance, they are reactive, and their performance is restricted to what they have learned. We included a DRL algorithm (Martinez-Baselga et al. 2023) in our platform and detected that it had room for improvement in scenarios where humans remained static. In these situations, it reacted late (possibly expecting motion from the human side). This is probably due to the fact that in the simulator it was trained, mostly dynamic obstacles were considered. Even though the performance could be probably enhanced with some refinements in the training process, it is an example of the challenges DRL faces nowadays. SHINE has collision restrictions and plans in advance, so it may adapt to any environments. The difference in performance is shown in Figure 12 (a) and Figure 12 (b).

**6.4.3 Zig-zag scenario** An MPC-based method gets trapped in local optima if the predictions do not meet the real behavior of the obstacles. We tested LMPCC (Brito et al. 2019) in a scenario where a pedestrian chooses one side to avoid the robot and then, unexpectedly, tries to avoid it through the other side, following a "zig-zag" motion. The result, as seen in Figure 13 (a), is that the robot tries to keep the same homology class, accelerating and curving its



**Figure 12.** Graphs of the result of a static scenario with Social DRL (a) and SHINE (b) planners. The figure follows the same representation as Figure 11.



**Figure 13.** Graphs of the result of a zig-zag scenario with LMPCC (a) and SHINE (b) planners. The figure follows the same representation as Figure 11.

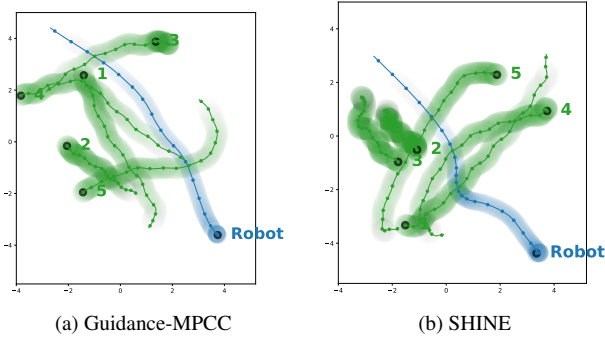
trajectory to avoid the human through the same side (it passes before the pedestrian). Our approach (Figure 13 (b)) replans in a different homology class to avoid the pedestrian to avoid the pedestrian through the other side (it passes after the pedestrian), resulting in a smoother and less curvy trajectory.

**6.4.4 Crowded scenarios** This experiment differed from the others in that we were not addressing a specific limitation of a method but rather assessing overall performance. We conducted the experiment in scenarios involving several moving pedestrians. Five participants were randomly selected and instructed to wander around the room freely. They were informed that a robot would also be moving in the room and that they should assume it would behave like another person.

To begin, participants were asked to walk around the room for one minute without the robot, allowing them to get accustomed to the environment. Following this, they repeated the exercise while sharing the space with the moving robot for five minutes. Afterward, each participant individually and separately evaluated the robot's motion qualitatively for one minute to avoid mutual influence.

This sequence was repeated four times, each time with a different motion planner controlling the robot. The planners tested in the experiment were Social Force, DRL, Guidance-MPCC, and SHINE, in that order. We recorded the positions of both the robot and the pedestrians, participants' evaluations of each planner, and a video of the experiment.

In the evaluations, all the participants assigned higher grades to SHINE than to Guidance-MPCC. Social Force and



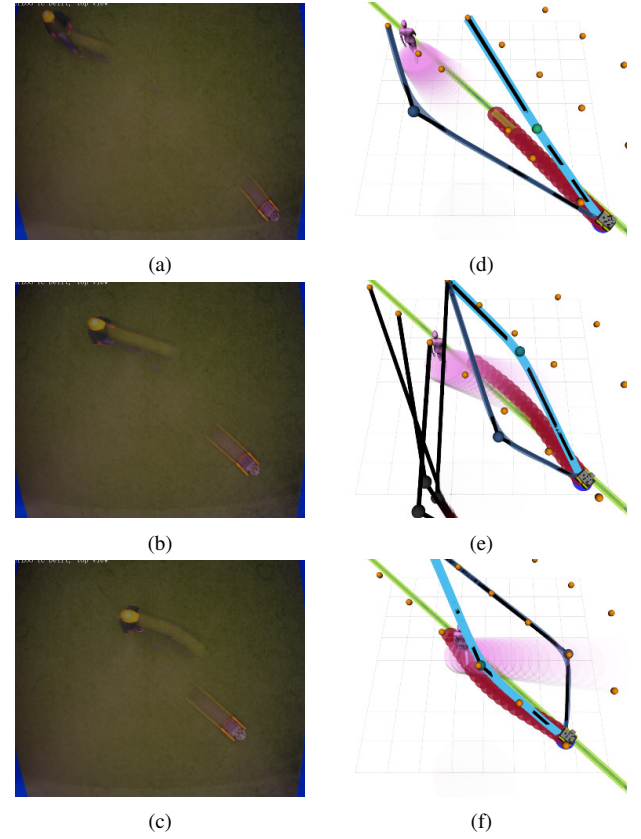
**Figure 14.** Graphs of the result of a crowded scenario with Guidance-MPCC (a) and SHINE (b) planners. The figure follows the same representation as Figure 11. Guidance-MPCC decides to pass before the first human it encounters (a), making him adapt its trajectory, while SHINE passes after him (b), so the resulting trajectory of the human is smooth.

DRL had a very reactive behavior that was not desirable for navigating around humans, even leading to some collisions.

We analyzed the behavior of SHINE in the experiments and compared it with the behavior of Guidance-MPCC in terms of the homology class chosen in Table 9. In every situation where the robot could choose among different homology classes, we annotated how many times it chose to avoid the corresponding obstacle through the right or left. Additionally, in situations that were not head-ons, we measured the number of times the robot chose to pass before or after the corresponding human. The table shows that there is a higher tendency of avoiding on the right side in both methods. This probably happens because the pedestrians participating in the experiment have a cultural tendency of choosing the right side. Nevertheless, when comparing the times the robot chose to pass before or after the obstacle, there is a clear difference. Guidance-MPCC prefers passing before other pedestrians, as it chooses the guidance trajectories where it may reach the goal faster while keeping high and constant velocities. Our network, however, has a clear preference of passing after the other humans, due to the fact that it is socially desirable to not interrupt other pedestrians' trajectories if they have already started them unless there is enough space and they are not walking fast. For example, Figure 14 (a) shows a scenario where the robot navigating with Guidance-MPCC explicitly maneuvers to accelerate and pass before the first human it encounters. Human 5, initially on a direct path, was forced to adopt a curvy trajectory, even though his intended trajectory was clear in advance. Figure 14 (b) shows how SHINE maneuvers to pass after the first human it faces, so the human does not have to modify its trajectory and it looks smooth, resulting in higher perceived comfort.

**Table 9.** Passing behavior of Guidance-MPCC and SHINE in the real-world experiment.

Method	Left	Right	Before	After
G-MPCC	0.437	0.563	0.596	0.404
SHINE	0.463	0.538	<b>0.329</b>	<b>0.671</b>



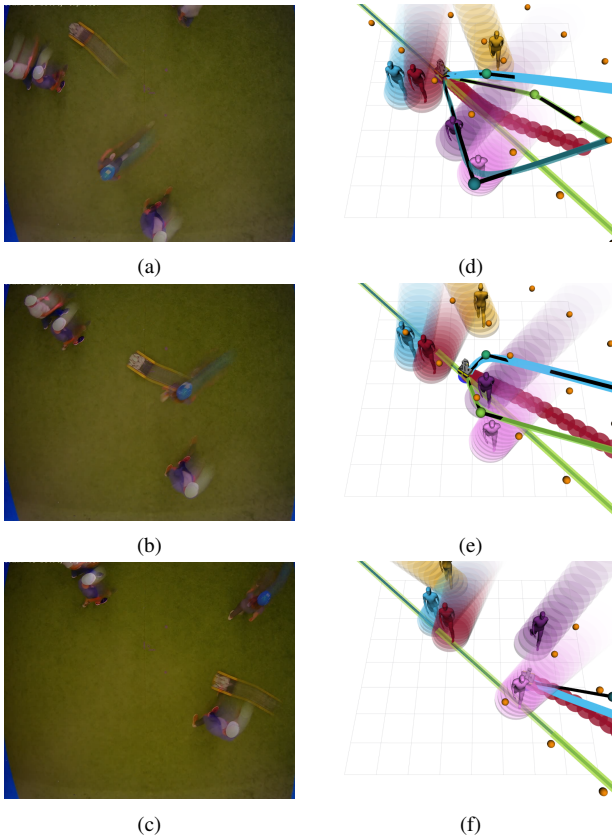
**Figure 15.** Video frames and visualization of a sequence where a robot adapts reactively to human intentions. The light green line that crosses the visualization is the reference path to be followed by the robot. The shadow with increasing transparency that starts in the human is its predicted trajectory. The orange dots are the possible subgoals to follow the reference paths. The colored lines that connect the robot and the subgoals are possible guidance trajectories, while the one highlighted with light blue is the one chosen. The red circles on the ground represent the optimized path computed by LMPCC.

**6.4.5 Scenarios analysis** An interesting behavior of our system may be observed in Figure 15, where there is a head-on scenario where a person changes the way of avoiding the robot (zig-zag scenario). At first, the human's intention is avoiding the robot on the right side, and the guidance chosen by the robot avoids the human on the right side too (light blue). Then, the human decides to avoid the robot through the left. The robot perceives his intentions and changes the homology class, following a smooth avoidance maneuver.

Another example of interesting behavior is shown in Figure 16. In that sequence, the robot faces a crossing with two humans at the same time, one of them with high velocity and the other one with low velocity. The robot chooses to pass after the pedestrian with high velocity, as its clear intention is passing as fast as possible, and to pass before the one with low velocity, who is not disturbed by the robot.

## 7 Discussion

The results of this work hold remarkable implications for the field of social navigation. One of the key implications is the ability to learn high-level discrete decisions from humans, unlike most of the learning-based social navigation approaches, which focus on imitating the low level behavior



**Figure 16.** Video frames and visualization of a sequence where the robot passes after a human with a high velocity and before a human with low velocity. The visualization is the same as in Figure 15.

of humans (like DRL or imitation learning). They do not have collision constraints, and their performance is limited to the scenarios and behaviors seen during training, which may not exhibit all the intricacies of the real world, being extremely sensitive to out-of-distribution data. Another problem that some learning-based approaches have is that they need a simulator to train the model, so their performance is limited to the realism of the simulations. With SHINE, however, in the worst case, the network will select a collision-free path that goes through a non-social homology class, still leading to a smooth and collision-free trajectory. Moreover, our work could be easily integrated with local optimization-based planners that include social costs in the optimization, achieving local social motion too. Therefore, our method is safer and easier to train compared to pure learning-based approaches, while we still incorporate learned social aspects to the navigation problem.

Our model has shown a remarkable adaptability to different environments and scenarios. In addition, the modularity design of the network allows the incorporation of more contextual information from the environment (as in Trajectron++ (Salzmann et al. 2020)).

SHINE provides a combination of global and local planning in dynamic environments. Unlike only reactive and local planners, it may change between homology classes in real time. This is specially important in crowded environments, as human decisions may change in time and the robot behavior should adapt to those changes.

The homology class selection module relies solely on the previous positions of obstacles and the robot and has been trained using real-world human trajectory data, implicitly accounting for possible interactions with humans and among humans themselves. Through this approach, SHINE implicitly couples prediction and planning, using the learned knowledge in the planning process. Although the local planner, LMPCC, does not integrate prediction and planning, it generates a local plan within that homology class, thereby accounting for human interactions at a global level.

The prediction module does not try to find an optimal homology class, as it would depend on the user-specified navigation objective of the robot and it may even not exist. The goal of the module is finding the homology class that is the most likely to be chosen by a human. Due to the natural multi-modality in human trajectory selection, achieving a perfect prediction accuracy is impossible, having scenarios where more than one homology class are equally good depending on specific human preferences. Nevertheless, this is not a problem for SHINE, as selecting any equally good homology class results in equally good high-level motion planning.

## 8 Conclusion

This work presented a novel motion planner for dynamic environments that is capable of making human-like high-level decisions to navigate. To this end, we trained a model that predicts the difference between the homology class of a given trajectory and the homology class chosen by a human. In particular, the planner first detects the possible homology classes of the trajectories in the environment. Then, it uses the trained prediction model to estimate the homology class that is the most likely to be chosen by a human, and navigates according to that decision. We showed that our planner learned social behaviors such as a tendency of passing behind other pedestrians or complex decisions to improve humans' comfort, and extensively evaluated it in both simulation and a lab experiment. We compared it with other social motion planners designed for social navigation and analyzed how it overcomes their limitations. In addition, it demonstrated a safe and smooth behavior in a crowded real-world environment. Furthermore, the prediction framework shows impressive results in predicting the homology class chosen by the humans, which could be used in other problems and open new research lines.

This work could present certain limitations if it is applied in environments where the humans behavior considerably varies from the one in the training data; and it does not consider distinctions between robots and humans as dynamic obstacles or the underlying topology of the map. These limitations present opportunities for improvement by adding more training data of heterogeneous situations or integrating additional map or multi-robot features into the network architecture, due to its a modular design.

Even though SHINE accounts for human future interactions in a high level, the local planner used, LMPCC, does not plan considering them. Future work could involve studying how to integrate a local planner that couples low level interactions with local planning. In addition, extending



the training data with datasets that involve other scenarios such as groups of people could expand and improve SHINE performance and adaptability with more kinds of social interactions.

### Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by MICIU/AEI/10.13039/501100011033 and ERDF/EU under grants (PID2022-139615OB-I00 and PRE2020-094415), from the Government of Aragón under grant DGA T45-23R, from the European Union's Horizon 2020 research and innovation programme under grant agreement No. (101017008), and from the European Union ERC, INTERACT, (101041863). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

### References

- Alonso-Mora J, Breitenmoser A, Beardsley P and Siegwart R (2012) Reciprocal collision avoidance for multiple car-like robots. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, pp. 360–366.
- Bai H, Cai S, Ye N, Hsu D and Lee WS (2015) Intention-aware online pomdp planning for autonomous driving in a crowd. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 454–460.
- Bandyopadhyay T, Won KS, Frazzoli E, Hsu D, Lee WS and Rus D (2013) Intention-aware motion planning. In: *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*. Springer, pp. 475–491.
- Bhattacharya S, Likhachev M and Kumar V (2011) Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In: *Robotics: Science and Systems*. pp. 27–30.
- Bhattacharya S, Likhachev M and Kumar V (2012) Topological constraints in search-based robot path planning. *Autonomous Robots* 33: 273–290.
- Biswas A, Wang A, Silvera G, Steinfeld A and Admoni H (2022) Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction (THRI)* 11(3): 1–24.
- Brito B, Floor B, Ferranti L and Alonso-Mora J (2019) Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters* 4(4): 4459–4466.
- Chen C, Hu S, Nikdel P, Mori G and Savva M (2020a) Relational graph learning for crowd navigation. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10007–10013.
- Chen C, Liu Y, Kreiss S and Alahi A (2019) Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In: *2019 international conference on robotics and automation (ICRA)*. IEEE, pp. 6015–6022.
- Chen Y, Liu C, Shi BE and Liu M (2020b) Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. *IEEE Robotics and Automation Letters* 5(2): 2754–2761.
- Chen YF, Liu M, Everett M and How JP (2017) Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 285–292.
- de Groot O, Ferranti L, Gavrila D and Alonso-Mora J (2023) Globally guided trajectory planning in dynamic environments. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 10118–10124.
- Ding Z, Liu J, Chi W, Wang J, Chen G and Sun L (2023) PRTIRL based socially adaptive path planning for mobile robots. *International Journal of Social Robotics* 15(2): 129–142.
- Domahidi A and Jerez J (2014) *FORCES Professional*. Embotech AG. URL <https://embotech.com/FORCES-Pro>.
- Everett M, Chen YF and How JP (2018) Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3052–3059.
- Everett M, Chen YF and How JP (2021) Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access* 9: 10357–10377.
- Fiorini P and Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research* 17(7): 760–772.
- Fox D, Burgard W and Thrun S (1997) The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1): 23–33.
- Francis A, Pérez-d'Arpino C, Li C, Xia F, Alahi A, Alami R, Bera A, Biswas A, Biswas J, Chandra R et al. (2023) Principles and guidelines for evaluating social robot navigation algorithms. *arXiv preprint arXiv:2306.16740*.
- Fridovich-Keil D, Ratner E, Peters L, Dragan AD and Tomlin CJ (2020) Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 1475–1481.
- Guo K, Wang D, Fan T and Pan J (2021) VR-ORCA: Variable responsibility optimal reciprocal collision avoidance. *IEEE Robotics and Automation Letters* 6(3): 4520–4527.
- Helbing D and Molnar P (1995) Social force model for pedestrian dynamics. *Physical review E* 51(5): 4282.
- Hu Z, Zhao Y, Zhang S, Zhou L and Liu J (2022) Crowd-comfort robot navigation among dynamic environment based on social-stressed deep reinforcement learning. *International Journal of Social Robotics* 14(4): 913–929.
- Ivanovic B, Harrison J and Pavone M (2023) Expanding the deployment envelope of behavior prediction via adaptive meta-learning. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 7786–7793.



- Jiang YQ, Chen BK, Wang BH, Wong WF and Cao BY (2017) Extended social force model with a dynamic navigation field for bidirectional pedestrian flow. *Frontiers of Physics* 12: 1–9.
- Kolur K, Chintalapudi S, Boots B and Mukadam M (2019) Online motion planning over multiple homotopy classes with gaussian process inference. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2358–2364.
- Kretzschmar H, Spies M, Sprunk C and Burgard W (2016) Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research* 35(11): 1289–1307.
- Lerner A, Chrysanthou Y and Lischinski D (2007) Crowds by example. In: *Computer graphics forum*, volume 26. Wiley Online Library, pp. 655–664.
- Lorente MT, Owen E and Montano L (2018) Model-based robocentric planning and navigation for dynamic environments. *The International Journal of Robotics Research* 37(8): 867–889.
- Martinez-Baselga D, de Groot O, Knoedler L, Riazuelo L, Alonso-Mora J and Montano L (2024) Repository of shine: Social homology identification for navigation in crowded environments. URL <https://github.com/dmartinezbaselga/SHINE>.
- Martinez-Baselga D, Riazuelo L and Montano L (2023) Improving robot navigation in crowded environments using intrinsic rewards. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 9428–9434.
- Matínez-Baselga D, Riazuelo L and Montano L (2023) Long-range navigation in complex and dynamic environments with full-stack S-DOVS. *Applied Sciences* 13(15): 8925.
- Mavrogiannis C, Balasubramanian K, Poddar S, Gandra A and Srinivasa SS (2022) Winding Through: Crowd navigation via topological invariance. *IEEE Robotics and Automation Letters* 8(1): 121–128.
- Mavrogiannis C, Baldini F, Wang A, Zhao D, Trautman P, Steinfeld A and Oh J (2023) Core challenges of social robot navigation: A survey. *ACM Transactions on Human-Robot Interaction* 12(3): 1–39.
- Mavrogiannis CI, Blukis V and Knepper RA (2017) Socially competent navigation planning by deep learning of multi-agent path topologies. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6817–6824.
- Mavrogiannis CI and Knepper RA (2019) Multi-agent path topology in support of socially competent navigation planning. *The International Journal of Robotics Research* 38(2-3): 338–356.
- McCammon S and Hollinger GA (2021) Topological path planning for autonomous information gathering. *Autonomous Robots* 45: 821–842.
- Mehr N, Wang M, Bhatt M and Schwager M (2023) Maximum-entropy multi-agent dynamic games: Forward and inverse solutions. *IEEE transactions on robotics* 39(3): 1801–1815.
- Missura M and Bennewitz M (2019) Predictive collision avoidance for the dynamic window approach. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8620–8626.
- Novosad M, Penicka R and Vonasek V (2023) CTopPRM: Clustering topological PRM for planning multiple distinct paths in 3D environments. *IEEE Robotics and Automation Letters* 8(11): 7336–7343.
- Pellegrini S, Ess A, Schindler K and 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*. IEEE, pp. 261–268.
- Penicka R and Scaramuzza D (2022) Minimum-time quadrotor waypoint flight in cluttered environments. *IEEE Robotics and Automation Letters* 7(2): 5719–5726.
- Penicka R, Song Y, Kaufmann E and Scaramuzza D (2022) Learning minimum-time flight in cluttered environments. *IEEE Robotics and Automation Letters* 7(3): 7209–7216.
- Peters L, Bajcsy A, Chiu CY, Fridovich-Keil D, Laine F, Ferranti L and Alonso-Mora J (2024) Contingency games for multi-agent interaction. *IEEE Robotics and Automation Letters*.
- Poddar S, Mavrogiannis C and Srinivasa SS (2023) From crowd motion prediction to robot navigation in crowds. *arXiv preprint arXiv:2303.01424*.
- Rösmann C, Hoffmann F and Bertram T (2017a) Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems* 88: 142–153.
- Rösmann C, Oeljeklaus M, Hoffmann F and Bertram T (2017b) Online trajectory prediction and planning for social robot navigation. In: *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, pp. 1255–1260.
- Rudenko A, Palmieri L, Herman M, Kitani KM, Gavrila DM and Arras KO (2020) Human motion trajectory prediction: A survey. *The International Journal of Robotics Research* 39(8): 895–935.
- Salzmann T, Ivanovic B, Chakravarty P and Pavone M (2020) Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, pp. 683–700.
- Siegwart R and Nourbakhsh IR (2011) *Introduction to Autonomous Mobile Robots*. 2 edition. The MIT Press.
- Siméon T, Laumond JP and Nissoux C (2000) Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics* 14(6): 477–493.
- Singamaneni PT, Bachiller Burgos P, Manso LJ, Garrell Zulueta A, Sanfeliu Cortés A, Spalanzani A and Alami R (2024) A survey on socially aware robot navigation: Taxonomy and future challenges. *The international journal of robotics research*.
- Sun M, Baldini F, Trautman P and Murphey T (2021) Move Beyond Trajectories: Distribution Space Coupling for Crowd Navigation. In: *Proceedings of Robotics: Science and Systems*. Virtual.
- Trautman P, Ma J, Murray RM and Krause A (2015) Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research* 34(3): 335–356.
- Van Den Berg J, Guy SJ, Lin M and Manocha D (2011) Reciprocal n-body collision avoidance. In: *Robotics Research: The 14th International Symposium ISRR*. Springer, pp. 3–19.
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł and Polosukhin I (2017) Attention is all you need. *Advances in neural information processing systems* 30.
- Vemula A, Muelling K and Oh J (2018) Social attention: Modeling attention in human crowds. In: *2018 IEEE international*

- Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4601–4607.
- Wakulicz J, Brian Lee KM, Vidal-Calleja T and Fitch R (2023) Topological trajectory prediction with homotopy classes. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 6930–6936.
- Wilkie D, Van Den Berg J and Manocha D (2009) Generalized velocity obstacles. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5573–5578.
- Xu C, Mao W, Zhang W and Chen S (2022) Remember intentions: Retrospective-memory-based trajectory prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 6488–6497.
- Yuan Y, Weng X, Ou Y and Kitani KM (2021) Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 9813–9823.
- Zhou B, Pan J, Gao F and Shen S (2021) Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics* 37(6): 1992–2009.
- Zhou Z, Zhu P, Zeng Z, Xiao J, Lu H and Zhou Z (2022) Robot navigation in a crowd by integrating deep reinforcement learning and online planning. *Applied Intelligence* 52(13): 15600–15616.