Socially Competent Navigation Planning by Deep Learning of Multi-Agent Path Topologies

Christoforos I. Mavrogiannis[†] Valts Blukis[‡] and Ross A. Knepper[‡]

***Abstract*— We present a novel, data-driven framework for planning socially competent robot behaviors in crowded environments. The core of our approach is a topological model of collective navigation behaviors, based on braid groups. This model constitutes the basis for the design of a human-inspired probabilistic inference mechanism that predicts the topology of multiple agents' future trajectories, given observations of the context. We derive an approximation of this mechanism by employing a neural network learning architecture on synthetic data of collective navigation behaviors. Our planner makes use of this mechanism as a tool for interpreting the context and understanding what future behaviors are in compliance with it. The planning agent makes use of this understanding to determine a personal action that contributes to the context in the most clear way possible, while ensuring progress to its destination. Our simulations provide evidence that our planning framework results in socially competent navigation behaviors not only for the planning agent, but also for interacting naive agents. Performance benefits include (1) early conflict resolutions and (2) faster uncertainty decrease for the other agents in the scene.**

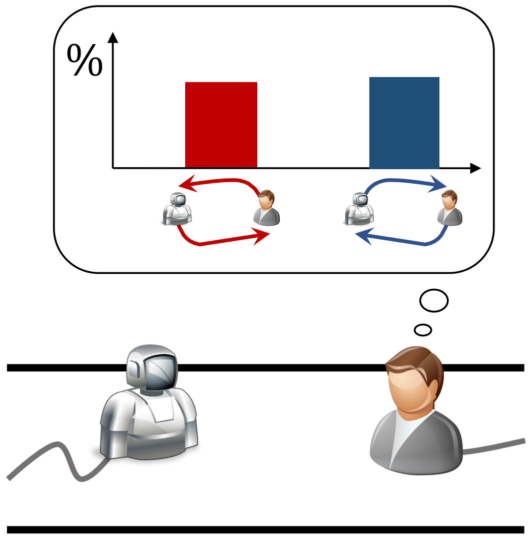

Fig. 1: A human and a robot navigating a hallway. From the perspective of the human, the behavior of the robot so far makes it unclear how the agents are going to avoid each other, yielding a high-entropy probability distribution over the type of avoidance.

I. Introduction

Over the past decades, robots are increasingly entering human environments, ranging from households and hospitals to city streets and malls. Naturally, the problem of navigation planning in crowded environments has received considerable attention. Navigating a crowded human environment is a particularly hard task for a robot, mainly due to the lack of formal rules regulating traffic, the lack of explicit communication among agents and the robot's imperfect sensing capabilities and inference models. Under such settings, existing planners often generate robot motion that feels unnatural from the perspective of human observers. Noteworthy examples include (1) the *freezing robot* problem [25] that occurs when the robot incorrectly assumes that all of the possible paths it could follow are blocked (2) the *reciprocal dance* problem [6], which occurs when the robot is unable to agree with a human on the passing side and (3) cases when the robot hinders, blocks or intersects planned human paths.

We argue that the aforementioned problems arise, in part, as a result of (1) the limited capability of the robot to *understand* the unfolding scene dynamics (2) its failure to act in a way that is compatible with these dynamics and (3) its inability to convey its intentions in a consistent, clear and timely fashion to human observers (see Fig. 1). In our previous work [15], we approached these issues by introducing a model that abstracts the scene dynamics into a topological signature; this model enables an agent to reason about the emerging collective behavior of all agents in a principled fashion and generate actions that are compatible with it. The approach was validated in simulation on a discretized board in a game-theoretic setup. In this paper, we build on this work and extend it in two significant directions: (a) we learn a predictive model of collective behavior from demonstrations of multi-agent navigation simulations and (b) we enable our models to handle continuous domains in time and space. Our framework enables agents to read and take into consideration the intentions of others at the decision making stage, which results in more comfortable navigation. This is illustrated in our simulation results that demonstrate accelerated consensus among agents regarding a collision avoidance strategy.

II. Related Work

The sophistication of human navigation has attracted the interest of several different scientific communities. Wolfinger [27] focused on the social aspect of navigation, observing that its smooth character relies on a form of trust that is established and reinforced among pedestrians. A class of works have focused on the development of models of crowd

[†]C. Mavrogiannis is with Sibley School of Mechanical & Aerospace Engineering, Cornell University, Ithaca NY, 14853, USA. Email: cm694@cornell.edu

[‡]V. Blukis and R. Knepper are with the Department of Computer Science, Cornell University, Ithaca NY, 14853, USA. Email: vb295@cornell.edu, rak@cs.cornell.edu

This material is based upon work supported by the National Science Foundation under Grant Nos. 1526035 and 1563705 and by the Air Force Office of Scientific Research. We are grateful for this support.

dynamics towards analyzing, understanding and simulating realistically crowd flows in different scenarios and contexts [8, 10, 16, 28], whereas another has focused on models for online trajectory prediction and tracking [2, 18].

The robotics community, incorporating the insights of studies from the aforementioned communities, has been working on the development of frameworks for integrated inference and planning for real-time robotic navigation of human environments. Under this perspective, the issue of human comfort has been of particular interest. To this end, Sisbot et al. [19] introduced a planner that made use of cost functions for measuring different aspects of human comfort, whereas a class of works has focused on learning models of human behavior towards acting in a socially compliant fashion [14, 25, 29]. Finally, another class of works, inspired by the collaborative nature of human navigation, have presented planners that incorporate models of multi-agent interactions and achieve oscillation-free collision avoidance [11, 26].

Our approach is unique in the sense that it augments several of the principled design directions of the literature with the introduction of new concepts and models from other disciplines. Specifically, our work is also motivated and inspired by the **collaborative** character of human navigation, as highlighted by Wolfinger [27]. We explicitly incorporate this observation into our framework through a novel **topological** representation, that makes use of the braid group [3] as a tool for abstracting the collective dynamics of a multi-agent scene. We employ this representation into the design of a **human-inspired** inference mechanism, supported by studies on human action interpretation [4], that enables an agent to reason about the collective behavior of multiple agents. We take a **data-driven** approach to learn a model of this mechanism by making use of a deep neural network architecture on trajectories extracted from multi-agent navigation simulations. Finally, we adopt an **information-theoretic** perspective to design a decision making policy that generates actions towards reducing uncertainty for everyone in the scene, thus yielding *socially competent* behaviors. Thanks to its topological foundation, our framework is generalizable to any environment geometry with any number of agents. Instead of making detailed trajectory predictions, it makes collective topology predictions that provide a more robust way of reasoning about the scene evolution. Furthermore, instead of generating reactions to individual observed behaviors, the planning agent makes explicit use of the realization that its behavior is part of the collective behavior of the system of agents.

III. Socially Competent Navigation

According to Wolfinger [27], the social order of human navigation relies on a high-level protocol, comprising two simple rules: (1) *people must behave like competent pedestrians* and (2) *people must trust copresent others to behave like competent pedestrians*. Although Wolfinger did not explicitly define competence, from the examples included in his work, we may deduce that he refers to a notion of *Social Competence*. The concept of Social Competence has been extensively studied in the field of Psychology from different perspectives and for different scenarios (for an extensive review see [17]). In multi-agent navigation, we may define social competence as:

> *The ability of an agent to perceive the context[1], analyze it and pick an action that appears to be compatible with it, according to a pattern of behavior that the agent assumes observing agents expect from him/her by having observed and analyzed the context themselves.*

According to Csibra and Gergely [4], humans tend to attribute *goals* to observed *actions* in a given context. Therefore, socially competent navigation behaviors should be indicative of agents' intentions and compatible with the context. In other words, socially competent agents should be cognizant of the fact that their behaviors implicitly communicate their intentions to any observing agents. The importance of implicit communication for human-robot interaction applications has lately been increasingly appreciated [5, 13, 24].

IV. Foundations

A set of agents $N = \{1, 2, ..., n\}$ navigate a workspace $\mathcal{Q} \subset \mathbb{R}^2$. The state of agent $i \in N$ is given by $q_i \in \mathcal{Q}$. Agent i starts from an initial position $q_i^s \in \mathcal{Q}$ and moves towards a destination q_i^d that lies in a destination region $D_i \subset \mathcal{Q}$. The path that agent i followed to reach D_i is a function $\xi_i : I \to \mathcal{Q}$, where $I = [0, 1]$ is a uniform time parameterization. This path is not known a priori; since the agents do not explicitly exchange information, they should be replanning frequently to account for the changes of the dynamic environment. Furthermore, the agents are assumed to be acting rationally, which in our context means that (1) they always aim at making progress towards their destinations and (2) they have no motive for acting adversarially against other agents (e.g. blocking their paths or colliding with them).

A. A Topological Model of Collective Navigation Behaviors

Let $Q = (q_1, \dots, q_n) \in \mathcal{Q}^n$ denote the state of the system of all agents. The system starts from a state $Q_s = (q_1^s, \dots q_n^s)$ and evolves to $Q_d = (q_1^d, \dots q_n^d)$ by the end of the execution. The collective behavior of all agents throughout the evolution of the scene may be described by a *system path* Ξ from the space of system paths $\mathcal{Z}$, comprising all possible system paths that start from Q_s and end at Q_d. The system path is a function $\Xi : I \to \mathcal{Q}^n \backslash \Delta$, where $\Delta = \{Q = (q_1, q_2, \dots, q_n) \in \mathcal{Q}^n : q_i = q_j \text{ for some } i \neq j \in N\}$ is the set of all system states with agents in collision. Naturally Δ splits the space of system paths $\mathcal{Z}$ into a set of classes of homotopically equivalent system paths. Each such class has topological properties that indicate a distinct collective behavior. To enumerate such classes of collective behavior but also to characterize system paths topologically, we employ the method of Mavrogiannis and Knepper [15] that makes use of the *Braid Group* [3]. In the following

[1] By context, we refer to information that is publicly available (e.g. the map), information that may be directly acquirable through sensing (e.g. agent trajectories) and information that may be acquired through standard inference processes (e.g. agent groups).

paragraphs, we provide a primer on braids and briefly recap this method.

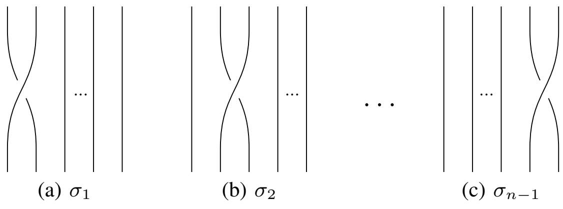

Fig. 2: The generators of the Braid Group B_n.

B. The Braid Group

Let $p : N \to N$ be a permutation in $Perm(N)$. A braid on $n \geq 1$ strands can be described as a system of n curves in $\mathbb{R}^3$, called the strands of the braid, such that each strand i connects the point $(i, 0, 0)$ with the point $(p(i), 0, 1)$ and intersects each plane $\mathbb{R}^2 \times t$ exactly once for any $t \in [0, 1]$. The set of all braids on n strands, along with the composition operation, form a group B_n. B_n can be generated from a set of $n-1$ elementary braids $\sigma_1, ..., \sigma_{n-1}$, called the generators of B_n (see Fig. 2), that satisfy the following *relations*:

$$\sigma_j \sigma_k = \sigma_k \sigma_j, \qquad |j-k| > 1, \tag{1}$$

$$\sigma_j \sigma_k \sigma_j = \sigma_k \sigma_j \sigma_k, \qquad |j-k| = 1. \tag{2}$$

Intuitively, a generator σ_i, $i \in \{1, 2, ..., n-1\}$, can be described as the pattern that emerges upon exchanging the ith strand (counted from left to right) with the $(i+1)$th strand, such that the initially left strand passes over the initially right one, whereas the inverse element σ_i^{-1} implements the same strand exchange, with the difference that the left strand passes under the right. An identity element, e, is a braid with no strand exchanges. Two braids $b_1, b_2 \in B_n$ may be composed through the composition operation $(\cdot)$, which is algebraically denoted as a product $b_1 \cdot b_2$. Geometrically, this composition results in the pattern that emerges upon attaching the lower endpoints of b_2 to the upper endpoints of b_1 and shrinking each braid by a factor of 2, along the t axis (see Fig. 3). Any braid can be written as a product of generators and their inverses. This product is commonly referred to as *braid word*.

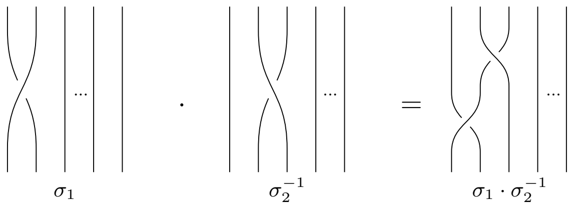

Fig. 3: The Composition $\sigma_1 \cdot \sigma_2^{-1}$ for the Braid Group B_n.

C. Representing Collective Navigation Behaviors as Braids

Fix a reference frame $\{F\}$ and denote by $f_x : \mathcal{Q}^n \to N$ a function that takes as input the system state $Q \in \mathcal{Q}^n$ and outputs a permutation $p \in Perm(N)$ corresponding to the arrangement of all agents in order of increasing x-coordinates with respect to $\{F\}$. As the agents move towards their destinations, they employ navigation strategies – maneuvers to avoid collisions. These result in a path of permutations $\mathcal{P} = (p_0, \dots, p_K)$, where $p_0 = f_x(Q_s)$ and $p_K = f_x(Q_d)$. We assume that $\mathcal{P}$ is the minimal permutation path from p_0 to p_K, i.e., $p_j \neq p_{j+1}$, for all $j = \{0, 1, \dots, K-1\}$ and that consecutive permutations are adjacent transpositions, i.e., they implement exactly one swap of exactly two adjacent elements. Therefore, due to continuity, a transition from the $(j-1)$th permutation p_{j-1} to the jth permutation p_j, implies the occurrence of an *event* τ_j, which we define as *the intersection of the x-projections of the paths of two agents that were adjacent in the permutation* p_{j-1}. Thus the event τ_j may be represented as an elementary braid $\tau_j \in \sigma_i^{\pm 1}, i \in \{1, ..., n-1\}$ and therefore the whole scene evolution may be represented as a braid – a product of the braids corresponding to the sequence of events $\tau = \tau_1 \tau_2 \dots \tau_K \in B_n$. This braid constitutes a topological characterization and abstraction of the system path. In the remainder of this paper, we will be referring to the sequence τ as the *topology* of the system path. Essentially, we model space of system path topologies $\mathcal{T}$ as the braid group, i.e., $\mathcal{T} := B_n$.

D. Inferring System Path Topologies from Context

Let us denote by M_t the *context* of the scene at time $t \in [0, 1]$. By context, we refer to information that is either publicly available (e.g. the map of the scene, points of interest, etc.), or directly acquirable through sensing (e.g. agents' state history) or indirectly acquirable through processing (e.g. agents' current arrangement $p \in Perm(N)$, inference about agents' destinations, their corresponding final ordering p_m, agents' groupings, etc.) during the time frame $[0, t]$. Assuming that by time $t \in [0, 1]$, a sequence of k events $\tau_1, \dots, \tau_k$ have already occurred, a model of the form $P(\tau_{k+1}, \dots, \tau_K | M_t)$ describes the probability of a future system path topology $\tau = \tau_{k+1} \dots \tau_K \in B_n$ given the context M_t. For simplicity, when referring to a prediction over future system path topologies, we will be using τ to refer to the sequence $\tau_{k+1} \dots \tau_K$.

The actions that agents select at each time step become part of the context, as they constitute information that may be directly acquirable by all agents through sensing. Therefore, having an understanding of what collective behaviors τ may be compatible with the context M_t, may allow an agent to contribute to it by executing actions that appear to be in compliance with the emerging collective behavior. In particular, an agent that is considering executing an action from a set of actions $\mathcal{A}$ may be able to understand how each action $a \in \mathcal{A}$ may reshape the belief of any observers, by simulating this action and computing $P(\tau | M_t, a)$. Using the chain rule of probability, this distribution may be factorized as:

$$P(\tau | M_t, a) = P(\tau_{k+1}, \tau_{k+2}, \dots, \tau_K | M_t, a) \tag{3}$$

$$= P(\tau_K | M_t, a, \tau_{k+1} \dots \tau_{K-1}) \tag{4}$$

$$\dots P(\tau_{k+1} | M_t, a)$$

Taking into consideration this distribution at the planning stage may enable an intelligent agent to make decisions that contribute to the context towards what appears to be the more likely or appropriate collective behavior, with respect to the current status of the context, M_t. In our scope, this is what

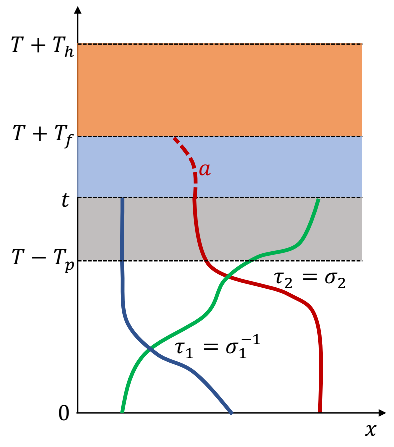

Fig. 4: Context and time flow around a planning step.

corresponds to *socially competent* behavior.

V. Learning Collective Navigation Behaviors

Little variations in agents' decision making and perception mechanisms may lead to significantly different correlations between the context and the topology. Thus, estimating the distribution $P(\tau|M_t, a)$ realistically with a closed-form model and without introducing over-simplifying assumptions is not a trivial task. For this reason, we adopt a data-driven approach to extract a model of the inference mechanism $P(\tau|M_t, a)$ from demonstrations of multi-agent navigation. We do so by training a model of the transition probabilities introduced in eq. (4). To the best of our knowledge, most publicly available pedestrian datasets either do not contain a sufficiently large volume of sufficiently diverse behaviors or are not in a format compatible to our setup. To overcome these complications, we generate a synthetic dataset of system paths, using the *Social Force* (SF) model [8]. In the following subsections, we describe the process of generating our dataset and detail our learning setup and architecture.

A. Generating a Dataset of Diverse Collective Behaviors

1) The Social Force Model: At its simplest form, the core of the Social Force model [8] is a dynamic artificial potential field, constructed by assigning repulsive potentials to agents, workspace boundaries and obstacles and attractive potentials to points of interest or destinations. Each agent is thus subjected to a resultant force that attracts it towards its destination and away from other agents, workspace boundaries or obstacles. Although the Social Force model may produce realistically looking pedestrian flows in simulation, it lacks a predictive component, which renders it as impractical for real-time robotics applications. However, for training purposes, the model enables us to generate a noise-free set of sufficiently diverse collective behaviors.

2) Experimental Setup: We randomly place a fixed number of agents n on the circumference of a circular or rectangular workspace. The agents are assigned destinations that enforce intense encounters (lying in the opposite side of the workspace) and move towards them by running individual instances of the social force model. The model parameters, as well as agents' initial positions and destinations are varied across experiments according to gaussian distributions.

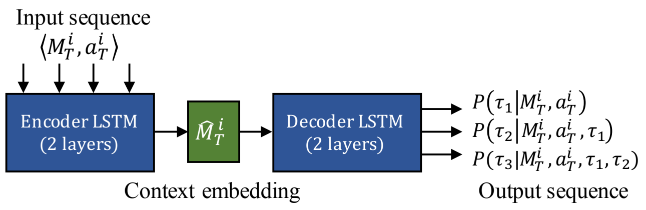

Fig. 5: Learning Architecture.

Experiments on the circular workspace loosely simulate pedestrian crossings in free areas such as atriums or parking lots, whereas experiments on the rectangular workspace resemble crossings in hallways. Each experiment is recorded as a waypoint representation of the system path with fixed time parametrization dt. The whole dataset is stored in a 4-dimensional tensor X of size $T_{max} \times N_{dof} \times n \times N_{exp}$, where T_{max} is the maximum number of time steps to destination taken by any agent in the dataset, N_{dof} is the size of agents' state ($N_{dof} = 2$ as we do not consider orientation) and N_{exp} is the total number of experiments.

B. Learning Setup

We split the acquired dataset into a set of N_x training examples; each example i is described by a feature tuple $\langle M_T^i, a_T^i \rangle$, where M_T^i is the context at time step $T \in \{1, \ldots, T_{max}\}$ and a_T^i is the action that agent i executed at that time step, both expressed with respect to frame F_i centered at the starting position of agent i, with y-axis pointing towards its destination. We consider the context M_T^i to be the system path of the time frame $(T - T_p, T]$, i.e., we make the assumption that the previous T_p time steps fully capture the context at time T. Similarly, we consider the action a_T^i to be the path that agent i followed in the frame $(T, T + T_f]$. Each example is labeled after the braid word $\tau \in B_n$ corresponding to the projection of the system path in the horizon $(T, T + T_h]$ onto the x-axis of frame F_i. Fig. 4 demonstrates the time flow around a training example.

C. Learning Architecture

Using the aforementioned setup, the goal of our learning algorithm is to extract models of the conditional probabilities of eq. (4), i.e., $P(\tau_1|M_T, a)$, ..., $P(\tau_K|M_T, a, \tau_1 ... \tau_{K-1})$, so that given an action $a \in \mathcal{A}$ and a system path topology τ of maximum braid length K, we can compute the probability $P(\tau|M_T, a)$. Essentially we need to produce the probability of an output sequence (braid word) given an input sequence (system path). This problem is essentially equivalent to a language translation task. Tasks of this form are effectively handled by sequence to sequence neural network models (see e.g. [21]). For this reason, we employ a sequence to sequence encoder-decoder learning architecture. The input sequence $\langle M_T^i, a_T^i \rangle$ is fed to an encoder Recurrent Neural Network (RNN), which produces an embedding vector $\widehat{M_T^i}$ that captures the expected future system path topology. The embedding vector is then fed to a decoder RNN that outputs estimates of $P(\tau_1|M_t^i, a_T^i)$, $P(\tau_2|M_t^i, a_t^i, \tau_1), \ldots, P(\tau_K|M_t^i, a_t^i, \tau_1, \ldots, \tau_{K-1})$. For the encoder and decoder RNNs we employ the Long Short-Term

Dataset	$X_{C,2}$	$X_{C,3}$	$X_{C,4}$	$X_{R,2}$	$X_{R,3}$	$X_{R,4}$
Training set	900k	1.6M	2.3M	800k	1.3M	1.8M
Test set	200k	400k	500k	200k	300k	400k

TABLE I: Generated dataset sizes (number of examples)

Dataset	τ	τ_1	τ_2	τ_3
$X_{C,2}$	0.93	0.93	1.00	1.00
$X_{R,2}$	0.99	0.99	1.00	1.00
$X_{C,3}$	0.71	0.77	0.93	0.98
$X_{R,3}$	0.78	0.88	0.89	0.97
$X_{C,4}$	0.45	0.62	0.74	0.88
$X_{R,4}$	0.56	0.81	0.78	0.81

TABLE II: Braid prediction accuracies for the whole topology τ and next, second and third events τ_1, τ_2, τ_3 respectively.

Memory (LSTM) architecture [9] due to its effectiveness in capturing long-term sequence dependencies. A schematic representation of our architecture is depicted in Fig. 5.

D. Implementation Details and Performance

We trained our models on 6 datasets of $100,000$ experiments each, labeled $X_{C,2}$, $X_{C,3}$, $X_{C,4}$, $X_{R,2}$, $X_{R,3}$, $X_{R,4}$ where the subscript denotes the number of agents involved ($n = 2, 3, 4$) and the type of workspace considered (C for circular, R for rectangular). From each dataset, we used 80,000 experiments for training and the rest for validation. As model parameters, we selected: $K = 3$, $T_p = 4$, $T_f = 3$, $T_h = 15$. Using this parametrization, we split the datasets into training examples and test examples as shown in table I. The examples were labeled as braids by using the Braidlab package [22]. As architecture parameters, we set both the encoder/decoder to use 2 LSTM layers and a hidden/cell state size of 80. The total number of trainable parameters is 216773. We train using a Dropout [20] of $p = 0.3$ after each layer. Our models were trained using the LSTM implementation of PyTorch [1]. We used the RMSProp [23] algorithm, considering $\alpha = 0.99$, a batch size of $10,000$ and an adaptive learning rate schedule, starting from $LR = 0.001$ and decreasing by a factor of 0.5 if no training loss improvement was observed after 3 epochs until it reached 0.00001. Every mini-batch was constructed with a proportional representation from each dataset, shuffling after every epoch.

The performance of our model in predicting future braid words is presented in tables II and III. Specifically, table II contains the accuracies of our models in predicting future topologies in total and per event for each per dataset, whereas table III contains the accuracies of our models per generator for each dataset, compared with a Prior baseline (each generator is assigned a probability equal to its frequency in the dataset) and Random Guessing (uniform probability for all generators). The accuracies for later time steps improve, because when the future topology τ contains less than 3 braid generators, all subsequent generators are trivially identity elements e i.e. no further crossings occur.

VI. Socially Competent Motion Generation

Our goal is to enable an autonomous agent to exhibit socially competent behavior in a multi-agent setting. From our perspective, this is equivalent to selecting actions that (1) are considered as appropriate within the state of the context M_t, (2) respect the personal space and the motion plans of others and (3) contribute progress towards the planning agent's destination.

Dataset	e	σ_1	σ_1^{-1}	σ_2	σ_2^{-1}	σ_3	σ_3^{-1}
$X_{C,2}$	0.98	0.64	0.76				
$X_{R,2}$	1.00	0.70	0.78				
Prior baseline	0.92	0.05	0.03				
Random guessing	0.33	0.33	0.33				
$X_{C,3}$	0.86	0.60	0.70	0.57	0.70		
$X_{R,3}$	0.90	0.87	0.89	0.85	0.87		
Prior baseline	0.47	0.13	0.13	0.14	0.12		
Random guessing	0.2	0.2	0.2	0.2	0.2		
$X_{C,4}$	0.77	0.51	0.59	0.55	0.61	0.44	0.53
$X_{R,4}$	0.94	0.78	0.81	0.62	0.78	0.78	0.81
Prior baseline	0.29	0.10	0.12	0.12	0.13	0.13	0.12
Random guessing	0.14	0.14	0.14	0.14	0.14	0.14	0.14

TABLE III: Per-permutation prediction accuracies for the next braid generator τ_1, compared against random guessing and guessing with probability proportional to the prior distribution (frequency of the generator).

A. Decision Making Policy

Building on our past work [15], we encapsulate the aforementioned specifications in a cost function $\mathcal{C} : \mathcal{A} \to \mathbb{R}$, defined as:

$$\mathcal{C}(a) = \lambda\mathcal{E}(a) + (1 - \lambda)\mathcal{H}(a) \tag{5}$$

where $\mathcal{E} : \mathcal{A} \to \mathbb{R}$ quantifies the *Efficiency* of an action $a \in \mathcal{A}$, $\mathcal{H} : \mathcal{A} \to \mathbb{R}$ quantifies the expected state of *Consensus* among agents over the emerging system path topology, upon executing the action in consideration, and λ is a weighting factor. We model Efficiency as the Euclidean distance between the position of the agent upon the execution of the action a and its destination. Consensus is modeled as the Information Entropy of the distribution over system path topologies $P(\tau|M_t, a)$:

$$\mathcal{H}(a) = -\sum_{\tau\in\mathcal{T}} P(\tau|M_t, a) \log P(\tau|M_t, a). \tag{6}$$

The higher the consensus cost, the more uncertain the evolution of the scene looks, as, from the definition of the Information Entropy, more outcomes-topologies will be more likely.

Thus, apart from making progress towards its destination, a socially competent agent has an incentive to actively reduce the uncertainty, by acting according to the context, in a way that reinforces everyones' belief regarding the emerging topology of the system path. The decision making policy for the socially competent agent can be formulated as:

$$a^* = \arg\min_{a\in\mathcal{A}} \mathcal{C}(a), \tag{7}$$

where $a^* \in \mathcal{A}$ is the action that contributes the maximal decrease of $\mathcal{C}$ in a given context, expressing an optimal compromise between progress to destination and consensus reinforcement, according to the weighting factor λ. Fig. 6 illustrates an example of reasoning about the future system path topology at planning time.

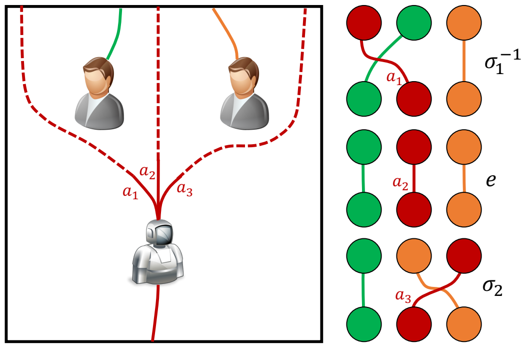

Fig. 6: The robot is reasoning about different actions leading to qualitatively distinct scene evolutions (left), corresponding to distinct system path topologies (right).

B. Generating a Set of System Path Topologies

In a scene with n agents, infinitely many, arbitrarily complex braids could be mathematically possible. However not all of them are likely to emerge. For computational and practical reasons, the planning agent concludes to a set $\mathcal{T} \subset B_n$ of likely topologies. To do so, the agent maintains a graph, called permutohedron, comprising nodes-permutations and edges-elementary braids (see Fig. 7). At planning time, the agent determines the permutation with respect to the x-axis of its body frame that corresponds to the current system state Q and derives the set of all possible future braids words of a given length.

C. Generating a Set of Actions

The planning agent is assumed to have access to a set of actions $\mathcal{A}$, comprising trajectories of a fixed number of time steps that are executable by its dynamics (in this paper, we do not incorporate dynamics, assuming that an agent may move towards any direction). The action set is generated offline by considering a set of time-parametrized paths. At planning time, the planner rejects the subset of $\mathcal{A}$ that is likely to lead to collisions with the environment or other agents. The actions in the collision-free set $\mathcal{A}_{cf}$ are evaluated with respect to the cost $\mathcal{C}$ and the best one is executed. This approach is inspired by the works of Green and Kelly [7] and Knepper et al. [12], which provide efficient algorithms and a deeper intuition on path sampling and collision checking.

D. Online Algorithm

Algorithm 1 presents our algorithm for Socially Competent Navigation (SCN). The Function `UpdateContext` incorporates the current system state Q to the context M_t. Next, the function `CollisionChecking` checks the action set for collisions and returns a collision-free subset $\mathcal{A}_{cf} \subseteq \mathcal{A}$. Subsequently, the function `GetTopologies` derives a set of likely topologies $\mathcal{T}$. Then, the function `ScoreTopologies` evaluates every topology in $\mathcal{T}$ given each action $a \in \mathcal{A}_{cf}$ and the context M_t by using our learned model $P(\tau|M_t, a)$ and returns a corresponding matrix of probabilities P. Finally, the function `MinimizeUtilityCost` evaluates all actions in $\mathcal{A}_{cf}$ with respect to the utility cost $\mathcal{C}$ and returns the action a^* that both contributes the best compromise between progress to destination and communication of compliance with the most

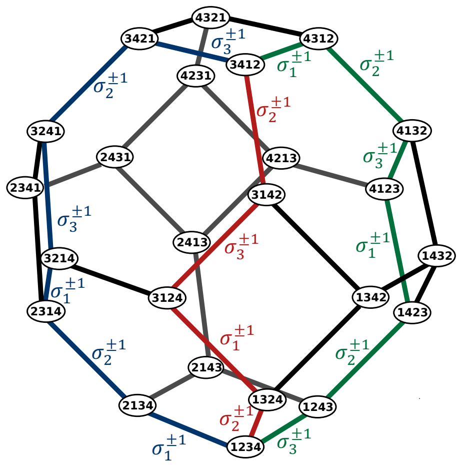

Fig. 7: Permutohedron of order 4: a graph with nodes representing all possible permutations of the set $N = \{1, 2, 3, 4\}$ and edges corresponding to transitions that are implementable with elementary braids (generators or their inverses).

likely system path topology at the given time. The algorithm runs until the agent reaches its destination, i.e., until the boolean variable $AtGoal$ becomes 1.

Algorithm 1 SCN($Q, \mathcal{A}, M_t, d, map, G, AtGoal, a$)

Input: Q – current system state; $\mathcal{A}$ – action set; d – agent's destination region; map; G – Permutohedron; $AtGoal$ – boolean variable signifying arrival at agent's destination; M_t – context
Output: a^* – action selected for execution

```
1: while ¬AtGoal do
2:     $M_t \leftarrow UpdateContext(Q, M_t)$
3:     $\mathcal{A}_{cf} \leftarrow CollisionChecking(\mathcal{A}, M_t, map)$
4:     $\mathcal{T} \leftarrow GetTopologies(Q, G)$
5:     $P \leftarrow ScoreTopologies(\mathcal{T}, \mathcal{A}_{cf}, M_t)$
6:     $a^* \leftarrow MinimizeUtilityCost(P, \mathcal{A}_{cf}, d, M_t)$
7: return $a^*$
```

VII. Evaluation

In our experimental evaluation, we aim to confirm (1) that an agent running SCN behaves as a socially competent pedestrian, contributing to other agents' certainty over future topologies and (2) that the presence of a socially competent agent improves the legibility of other agents' behavior.

A. Experimental Setup

We test our planning algorithm in 50 simulated scenarios involving 3 and 4 agents navigating a circular workspace. Each scenario is defined as a tuple (Q_s, Q_d), where Q_s was defined by placing each agent uniformly at random on the circumference of the workspace (see Fig. 8) and Q_d corresponds to points diametrically opposed to Q_s. The scenarios were deliberately designed to reinforce *intense* agent encounters.

For each scenario, we conducted 4 different experiments: (1) experiment `4SF`, where all agents are running an individual instance of the social force model, (2) experiment `1SCNv3SF`, where agent #1 is running our SCN algorithm

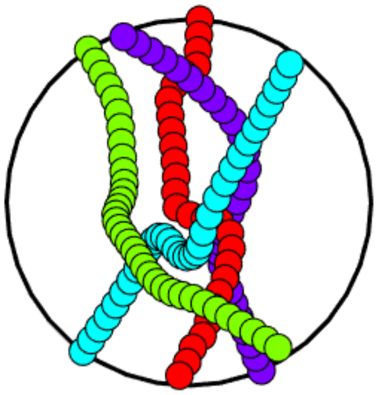

Fig. 8: Swept volumes of 4 agents navigating a circular workspace. The red agent runs SCN whereas the rest of the agents run a separate instance of the social force model.

and the other 3 agents are running the social force, (3) experiment 3SF, where we remove agent 1 and only simulate the remaining 3 agents using the social force model and (4) experiment 1SCNv3SF-1SCN that is essentially a playback of 1SCNv3SF with the trajectory of agent 1 (SCN) excluded (treated as invisible). Note that for the first 5 time steps, the SCN agents switch to SF in order to collect enough history to bootstrap the learning algorithm

For the evaluation stage, we train a separate evaluation model of the form $P(\tau|M_T^i)$ that predicts future topologies τ from the perspective of agent i based on the context M_T^i at time T. Both M_T^i and τ are expressed with respect to frame F_i for agent i (see section V-B). We use exactly the same model architecture and training procedure as detailed in section V, but with a shorter input sequence that excludes the action a. This model achieves similar accuracy to $P(\tau|M_T, a)$ (see Table III).

The action set for the agent running SCN comprised a collection of 31 time-parametrized ($dt = 0.2sec$, speed 1.2 m/sec) straight line path segments of 3 waypoints each, covering π rads, whereas the weighting factor λ was set to $\lambda = 0.6$. For reference, the parameters for the agents running instances of SF were selected as $v_{max} = 1.7m/sec$, $c = 1$, $\phi = 100°$, $R = 0.2m$, $\sigma = 0.5m$, $\tau_a = 0.4s$, $V^0_{\alpha\beta} = 20m^2/sec^2$, $U_{aB} = 10m^2/sec^2$, $v^0_a = 1.6m/sec$ and we kept the same time parametrization $dt = 0.2sec$.

B. Results

To demonstrate the benefits of the SCN algorithm for multi-agent navigation scenarios, we measure its effects in the behavior of other agents. More specifically, in each setup, we record (a) the time to destination (Fig. 9) and (b) the evolution of the entropy of the distribution $P(\tau|M_T^i)$ (Fig. 10), both averaged over the same three SF agents (agents #2, #3, #4) per experiment.

Fig. 9 shows that the 1SCNv3SF setup (Fig. 8 depicts an example execution) achieved the fastest average time to destination, as a result of SCN's consistently competent behavior. Student's t-test yields a p-value < 0.001 indicating a highly significant improvement, compared to 4SF and 3SF (Fig. 9). Note that the time to destination for the SCN agent itself was excluded from this test, indicating that the *three*

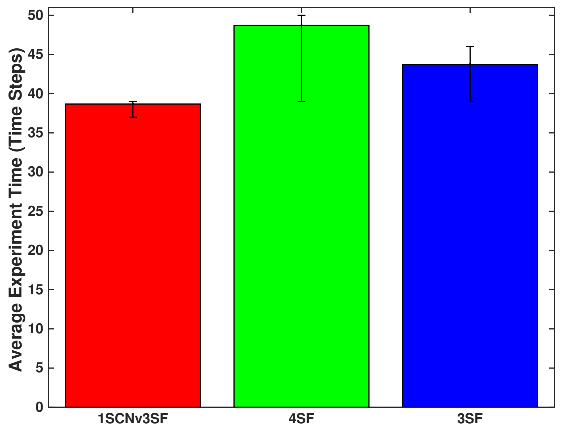

Fig. 9: Average time to destination per experiment per agent (excluding agent #1). The error bars indicate 25th and 75th percentiles over 50 experiments. 1SCNv3SF is shown to terminate faster than both 4SF and 3SF, (according to a Student's t-test with p-value < 0.001).

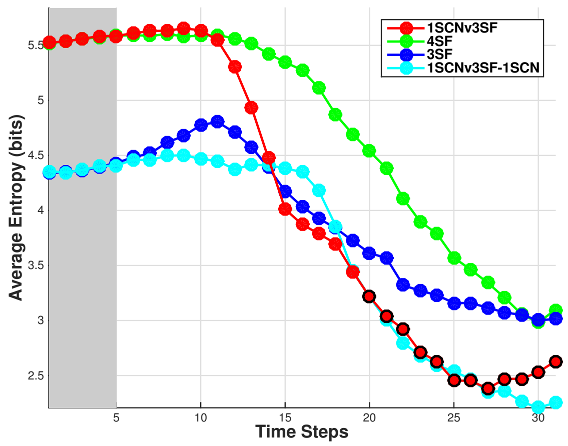

Fig. 10: Entropy profiles averaged across experiments and agents (excluding agent #1). The black circles indicate timesteps where the entropy measured for 1SCNv3SF is lower than for 3SF with statistical significance (Student's t-test, with p-value ≤ 0.022). The gray area in the plot corresponds to the first time frame T_p, during which the agent running SCN was moving efficiently and observing the context.

SF agents become more efficient thanks to the presence of one SCN agent.

In Fig. 10, at time zero, we see that the entropy of a uniform distribution in B_4 (the braid group with $n = 4$ strands) is naturally higher than the ones in B_3 simply because of the higher baseline penalty for the probability mass being spread over more discrete possibilities. As we noted earlier, during the first five time steps, SCN agents in 1SCNv3SF run SF in order to collect enough data (grey box in Fig. 10). Henceforth, they switch to SCN, which results in a precipitous drop of the average entropy that continues until it drops below both baselines. In particular, in the time frame $[25, 31]$ the entropy in 1SCNv3SF drops significantly below the entropy of 3SF, according to a Student's t-test, with p-value ≤ 0.022.

One could object that the SCN agent is an integral part of the braid, and it is therefore unsurprising that a socially competent agent reduces the system entropy. To measure the effect on the entropy of the other three agents alone, we introduce one additional baseline, `1SCNv3SF-1SCN`. This result shows the entropy of the system path for the three SF agents *after removing the SCN agent*. This result shows that the reduced entropy is not due to the direct contribution of SCN alone. Rather, the three SF agents are *themselves* behaving in a more orderly fashion in the presence of the SCN agent. This result suggests that in acting in a socially-competent manner, *SCN increases the social competence of SF agents as well*. For clarity, note that the graph in Fig. 10 terminates before all of the agents have had an opportunity to quiesce at their goals. At that point, all entropies converge to zero. However, the benefit of social competence in terms of reduced time and confusion is achieved long before.

VIII. Conclusion

We presented a planning framework for navigation in crowded environments. The foundation of our approach is a topological representation of the collective behavior of a set of agents, based on braids [3]. This representation forms the basis for the design of an inference mechanism that predicts the topology of the future trajectories of a set of agents, given the context of a scene. A model of this mechanism was extracted in a data-driven fashion by employing a deep neural network architecture on synthetic data generated through the use of the Social Force model [8]. The inference mechanism serves as a means of understanding how the agent's behaviors might affect the observing others. This enables it to select behaviors that are *socially competent*, i.e., constitute the best response to the context. We conducted a set of simulated experiments that provided us with statistically significant evidence suggesting that our framework results in collective behaviors that simplify the planning problem for everyone in the scene. This is reflected in the behavior of other agents: systems of agents containing an agent running our algorithm achieved significantly reduced average time to destination and were able to get a clear topological understanding of the scene evolution significantly faster, as shown in the average entropy profiles of the agents *not* running our model. Future work involves learning a model of system path topology prediction from human pedestrian data, experimental evaluation on a social robot platform and a study to assess its interactions with humans.

References

[1] Sam Gross Adam Paszke and Soumith Chintala. PyTorch. https://github.com/pytorch/pytorch. Accessed: [01/28/2017].

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[3] Joan S. Birman. *Braids Links And Mapping Class Groups*. Princeton University Press, 1975.

[4] G. Csibra and G. Gergely. 'Obsessed with goals': Functions and mechanisms of teleological interpretation of actions in humans. *Acta Psychologica*, 124(1):60–78, 2007.

[5] Anca D. Dragan and Siddhartha Srinivasa. Integrating human observer inferences into robot motion planning. *Auton. Robots*, 37(4):351–368, 2014.

[6] Franck Feurtey. Simulating the collision avoidance behavior of pedestrians. *Master's Thesis*, 2000.

[7] Colin Green and Alonzo Kelly. Toward optimal sampling in the space of paths. In *13th Int. Symposium of Robotics Research*, 2007.

[8] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, 1995.

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

[10] Ioannis Karamouzas, Brian Skinner, and Stephen J. Guy. Universal power law governing pedestrian interactions. *Phys. Rev. Lett.*, 113: 238701, 2014.

[11] Ross A. Knepper and Daniela Rus. Pedestrian-inspired sampling-based multi-robot collision avoidance. In *RO-MAN*, pages 94–100. IEEE, 2012.

[12] Ross A Knepper, Siddhartha S Srinivasa, and Matthew T Mason. Toward a deeper understanding of motion alternatives via an equivalence relation on local paths. *The International Journal of Robotics Research*, 31(2):167–186, 2012.

[13] Ross A. Knepper, Christoforos I. Mavrogiannis, Julia Proft, and Claire Liang. Implicit communication in a joint action. In *12th ACM international conference on Human-robot interaction*, 2017.

[14] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

[15] Christoforos I. Mavrogiannis and Ross A. Knepper. Decentralized multi-agent navigation planning with braids. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[16] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proc. of the Nat. Academy of Sciences*, 108(17):6884–6888.

[17] Douglas W Nangle, David J Hansen, Cynthia A Erdley, and Peter J Norton. *Practitioner's guide to empirically based measures of social skills*. Springer Science & Business Media, 2009.

[18] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc J. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, pages 261–268. IEEE Computer Society, 2009.

[19] Emrah Akin Sisbot, Luis Felipe Marin-Urias, Rachid Alami, and Thierry Siméon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

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

[22] Jean-Luc Thiffeault and Marko Budišić. Braidlab: A software package for braids and loops, 2013–2016. URL http://arXiv.org/abs/1410.0849. Version 3.2.1.

[23] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[24] Pete Trautman. Breaking the human-robot deadlock: Surpassing shared control performance limits with sparse human-robot interaction. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2017.

[25] Peter Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation. *Int. J. of Robotics Res.*, 34(3):335–356, 2015.

[26] Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Reciprocal *n*-body collision avoidance. In *ISRR*, pages 3–19, 2009.

[27] Nicholas H. Wolfinger. Passing Moments: Some Social Dynamics of Pedestrian Interaction. *Journal of Contemporary Ethnography*, 24(3): 323–340, 1995.

[28] Bolei Zhou, Xiaoou Tang, and Xiaogang Wang. Learning collective crowd behaviors with dynamic pedestrian-agents. *International Journal of Computer Vision*, 111(1):50–68, 2015.

[29] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Proc. of the International Conference on Intelligent Robots and Systems*, 2009.