Moving in a Crowd: Safe and Efficient Navigation Among Heterogeneous Agents

Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and **Maria Gini**
Department of Computer Science and Engineering
University of Minnesota
200 Union St SE, Minneapolis MN 55455
{godoy, ioannis, sjguy, gini}@cs.umn.edu

Abstract

Multi-agent navigation methods typically assume that all agents use the same underlying framework to navigate to their goal while avoiding colliding with each other. However, such assumption does not hold when agents do not know how other agents will move. We address this issue by proposing a Bayesian inference approach where an agent estimates the navigation model and goal of each neighbor, and uses this to compute a plan that minimizes collisions while driving it to its goal. Simulation experiments performed in many scenarios demonstrate that an agent using our approach computes safer and more time-efficient paths as compared to those generated without our inference approach and a state-of-the-art local navigation framework.

1 Introduction

The safe and efficient navigation of agents is critical in the deployment of robots in real world applications such as multi-robot exploration, task allocation and search and rescue. This problem is challenging especially when the environment is populated with unknown moving agents, as the uncertainty about their future motions prevents the utility of pre-computing a collision-free and time-efficient path.

Many approaches for decentralized navigation in multi-agent environments have been proposed. However, they typically assume that all agents use the same navigation method, which does not necessarily hold in partially known environments. When there is uncertainty on the dynamics of the environment, an agent should be extra cautious to reduce the risk of collisions when moving to its goal. However, this behavior could also translate into unnecessarily long paths, resulting in inefficient motion. This limits the agent's ability to compute safe but also efficient paths.

In this work, we address this problem. Assuming a finite set of possible navigation models (such as the current most popular multi-robot navigation approaches) and goals in an environment, we use a Bayesian inference approach to estimate the probability that each of the agent's neighbors uses a specific model and moves to one of the goals, based on observations of its motion. Once a model and goal are estimated for a neighbor, it can be used to predict its future motion along with its neighborhood interactions. This allows for the computation of a safe path that takes the agent closer to its goal.

This work makes three main contributions. First, we propose a Bayesian method to infer the models and goals of the agents in a partially known environment. Second, we propose an action planning technique that incorporates the inferred motions and goals to compute collision-free and time-efficient paths. Third, we experimentally validate our approach and show that it leads to safer and more efficient motions in a variety of scenarios as compared to a state-of-the-art collision avoidance framework [van den Berg *et al.*, 2011], and to planning without estimation of the models and goals.

2 Related Work

The multi-agent navigation problem has been extensively studied in robotics, AI, and the crowd simulation community. Over the years different approaches have been proposed, including reactive techniques based on artificial potential fields [Khatib, 1986; Karamouzas *et al.*, 2014], social force models that use a mixture of physical and psychological forces [Helbing *et al.*, 2000; Pelechano *et al.*, 2007], graph-based techniques [Luna and Bekris, 2011; Solovey *et al.*, 2015; Sharon *et al.*, 2015; Cirillo *et al.*, 2014] and learning-based approaches [Melo and Veloso, 2011; Martinez-Gil *et al.*, 2015]. Recently, approaches that plan directly in the velocity space [Fiorini and Shiller, 1998] have gained a lot of popularity due to their robust nature and the guarantees that they offer about collision freeness. Among the most popular approaches is the ORCA framework that selects new collision-free velocities for the agents by solving a low dimensional linear program [van den Berg *et al.*, 2011]. ORCA has been extended to simulate agents that have different personalities [Guy *et al.*, 2011], account for uncertainty [Kim *et al.*, 2014], as well as agents that can quickly reach their goals by implicitly coordinating with each other [Godoy *et al.*, 2016]. However, unlike our approach, all the aforementioned multi-agent techniques assume that the agents use the same underlying method to navigate through the environment.

Most closely related to our work is the approach of Choi et al. [2014] that uses a reactive Gaussian-based motion controller to address the issue of safe robot navigation in human populated environments. [Trautman *et al.*, 2015] explores robot navigation in dense pedestrian crowds to predict pedes-

trian trajectories and achieve cooperative motion. The complexity of these approaches limits their ability to predict trajectories of more than 10 dynamic entities, in order to have real-time operation. Instead, we aim at computing both safe and time-efficient trajectories in simulated environments with up to 100 agents in real time.

Finally, prior work has also addressed interactions with heterogeneous agents based on Bayesian learning (see, e.g. [Barrett and Stone, 2015; Barrett *et al.*, 2013]). In particular, Barrett et al. [2015] used a Bayesian framework to learn the types of other agents in the Robocup domain. However, in this domain, agents can only move according to a given set of rules, and only interact with agents of known types. In contrast, in our multi-agent navigation domain, the agents can assume both known and unknown types.

3 Problem Formulation

In this paper, we address the problem of safely navigating an agent to its goal position in a time-efficient manner through environments that are populated by unknown heterogeneous agents. More formally, we are given an agent α modeled as a disk with radius r that has to reach a goal $\mathbf{g}$. At each time instant t, we denote the agent's position as $\mathbf{p}^t$, and its velocity as $\mathbf{v}^t$ that is subject by a maximum speed $v^{\max}$. Furthermore, α has a preferred velocity $\mathbf{v}^{\text{pref}}$, indicating the agent's desired direction of motion and speed. The environment for agent α is defined as a 2D virtual or physical space that includes other n (initially unknown) heterogeneous agents $A_1 \dots A_n$. Similar to α, each agent A_i has a radius r_i, a position $\mathbf{p}_i$, a velocity $\mathbf{v}_i$ and a goal position $\mathbf{g}_i$.

The task is then twofold: first, determine how the neighbors of α navigate in the environment, and second, use this information to steer α to its goal, avoiding collisions with the n heterogeneous agents while minimizing its travel time.

Assumptions. We assume that a fixed number of goal locations $\mathcal{G}$ exist that each agent A_i can choose from. Furthermore, a set $\mathcal{M}$ of different navigation models is given and each agent A_i is only allowed to use one of these models throughout a simulation. Finally, we assume that α has a partial knowledge about the environment in which it navigates. Specifically, α can sense the positions and velocities of its nearby agents, $NN_\alpha \subset A$, located inside a limited sensing range, but it is not aware of the goals of its neighbors and the navigation models that they employ.

Proposed approach. We propose a Bayesian inference approach to estimate the probability that an agent in A uses one of the $|\mathcal{M}|$ specified navigation models while moving towards one of the $|\mathcal{G}|$ known goals in the environment. With an estimate computed at each timestep of the model and goal of each agent A_i, α can plan over a set of possible actions to minimize the risk of collisions and generate time-efficient motions.

The agent α can consider two sets of actions. It can either optimize over a set of different preferred velocities while keeping its navigation parameters fixed, or plan over a set of different navigation models while having a fixed goal-oriented preferred velocity with a magnitude equal to $v^{\max}$. See Section 5 for more details.

As soon as an optimal action a^* is chosen, it is mapped to a new velocity $\mathbf{v}^{\text{new}}$ using an anticipatory collision avoidance method based on the ORCA navigation framework [van den Berg *et al.*, 2011]. While in ORCA each pair of agents involved in a potential collision shares the effort of averting the collision, in our approach, the agent α takes full responsibility of resolving a collision due to the presence of heterogeneous neighbors.

Algorithm 1 General approach

```
1: initialize simulation
2: while not at the goal do
3:    for all i ∈ NN_α do
4:       observe i velocity and position
5:       Pred_i ← NeighborInference(i)
6:       PredSet ← PredSet ∪ Pred_i
7:    end for
8:    a* ← ActionSelection(PredSet)
9:    v^new ← CollisionAvoidance(a*)
10:   p^{t+1} ← p^t + v^new · Δt
11: end while
```

A pseudocode of our approach can be seen in Algorithm 1. While agent α has not reached its goal, it estimates the model and goal for each sensed neighbor (line 5), using Bayesian inference. It then uses this information to determine a new action which takes into account the predicted motion of each neighbor (line 8), according to its estimated model. The action is then given as an input to the collision avoidance routine, to compute a new velocity that is collision-free (line 9). After moving using its computed velocity, the cycle repeats until α reaches its goal.

3.1 Navigation models

We consider the following navigation models for the agents in A, which are representative of existing classes of multi-agent navigation:

- Personality models: Three of the models are based on the ORCA navigation framework [van den Berg *et al.*, 2011]. These models have different sets of parameter values representing social behaviors or personalities, as proposed in [Guy *et al.*, 2011]. We consider three types of agents: shy, aggressive, and tense.
- Social force model: In the work of Helbing et al. [2000], the behavior of each agent is modeled as a collection of forces. Each agent has an attractive force that steers it to its goal. In addition, distance-dependent repulsive forces are exerted on the agent from its nearby neighbors and static obstacles. Thus, the social force model is a purely *reactive* model as opposed to the *anticipatory* nature of ORCA and its variants.
- Non-reactive: In this model, agents do not attempt to avoid or resolve collisions, i.e. they are non-reactive and just follow their goal oriented trajectories.

We also consider agents that might not use any of the specified navigation models, for example, in case of malfunction, and can have random goals in the environment and random navigation parameters. We note that our approach can be easily extended to account for new navigation models.

4 Bayesian Model Inference

We use a Bayesian approach for α to estimate the navigation model and goal position of each of its neighbors. In particular, for each neighbor and at each timestep t, we compute the position it will move to in $t+1$ assuming each of the possible model/goal combinations (Fig. 1a).

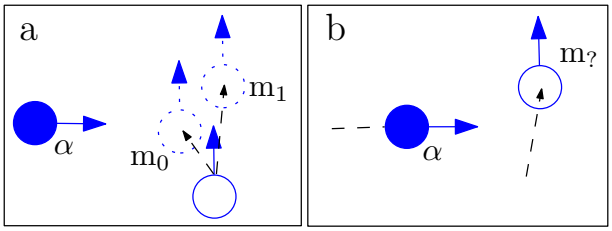

Figure 1: Example of the Bayesian model inference, assuming two navigation models and a single goal. (a) α predicts the position of its neighbor in the next timestep, using each of the two models m_0 and m_1. (b) After the agents move, α estimates its neighbor's true model based on its new position.

At time $t+1$, α observes the new position of each neighbor (Fig. 1b), and computes the likelihood of the observed position given each of the possible models, assuming Gaussian noise in their computed velocities.

$$P(\mathbf{p}_i^t | m_j \wedge \mathbf{g}_k) \sim \mathcal{N}(\mathbf{p}_i^t; \mu, \sigma), \tag{1}$$

$\forall\, m_j \in \mathcal{M}$ *and* $\forall\, \mathbf{g}_k \in \mathcal{G}$. To compute the likelihood that the neighbor's position was not generated by any model in $\mathcal{M}$, i.e. the neighbor uses an 'unknown' model, we use the conjunction of the complements of all the model likelihoods, which is equivalent to their multiplication (Eq. 2):

$$P(\mathbf{p}_i^t | m_u \wedge \mathbf{g}_u) = \prod_{\substack{j=1 \\ k=1}}^{|\mathcal{M}|\cdot|\mathcal{G}|} (1 - P(\mathbf{p}_i^t | m_j \wedge \mathbf{g}_k)) \tag{2}$$

Once the likelihood values have been computed for all model/goal pairs and a possible unknown model, the posterior probabilities are computed using a uniform prior in the first observation and the previous posterior as the prior in the subsequent observations (Eq. 3). This represents the initial uncertainty of the agent with respect to its neighbors' models and goals, and as more evidence is incorporated, the true model becomes more likely.

$$P(m_j \wedge \mathbf{g}_k | \mathbf{p}_i^t) = \frac{P(\mathbf{p}_i^t | m_j \wedge \mathbf{g}_k) \cdot P(m_j \wedge \mathbf{g}_k | \mathbf{p}_i^{t-1})}{\sum_{j'=1}^{|M|} \sum_{k'=1}^{|G|} P(\mathbf{p}_i^t | m_{j'} \wedge \mathbf{g}_{k'}) \cdot P(m_{j'} \wedge \mathbf{g}_{k'} | \mathbf{p}_i^{t-1})} \tag{3}$$

4.1 Model Selection

Model Estimation. After computing the posterior probabilities for each model at each timestep, α can estimate the true model of each neighbor. While choosing the model with the maximum posterior would be reasonable, in practice there might be many models that move the agent to the same or a very similar position. This translates into similarly high values for the posteriors. For example, a neighbor agent being pushed by a crowd might be forced to escape from the crowd to avoid collisions, regardless of the model being used.

Algorithm 2 Model Selection for neighbor i

for $k = 1, \ldots, |\mathcal{G}|$ **do**
  for $j = 1, \ldots, |\mathcal{M}|$ **do**
    $ComputeLikelihood(i, m_j, \mathbf{g}_k)$ (Eq. 1)
    $ComputePosterior(i, m_j, \mathbf{g}_k)$ (Eq. 3)
  end for
end for
$ComputeLikelihood(i, m_u, \mathbf{g}_u)$ (Eq. 2)
$ComputePosterior(i, m_u, \mathbf{g}_u)$ (Eq. 3)
for $k = 1, \ldots, |\mathcal{G}|$ **do**
  for $j = 1, \ldots, |\mathcal{M}|$ **do**
    if $\frac{P(m_j \wedge \mathbf{g}_k | \mathbf{p}_i^t)}{\max P(m_j \wedge \mathbf{g}_k | \mathbf{p}_i^t)} > (1 - \gamma)$ **then**
      $C_{mod} \leftarrow C_{mod} \cup (m_j, \mathbf{g}_k)$
    end if
  end for
end for
$Pred_i \leftarrow$ *randomly chosen* $(m_j, \mathbf{g}_k) \in C_{mod}$

To address this, we consider a set, C_{mod}, of candidate models for a neighbor that contains all models whose posterior probabilities are within a threshold γ away from the model with the maximum posterior. As the simulation progresses and α obtains more observations of how each neighbor moves, its true model should be more easily differentiated from the other candidate models, and the size of C_{mod} should decrease and finally converge to the true model.

Motion Prediction. While our model estimation may suggest that multiple models fit a neighbor's observations, we must choose one model for predicting its future trajectory. To do this, we select a single model, denoted $Pred_i$, randomly from C_{mod}. The likelihood of each candidate model being chosen is proportional to their respective normalized posterior values. Algorithm 2 shows the model selection procedure.

5 Action Planning

Once a model and goal have been estimated for each neighbor, α can use this information to predict their future motions along with their interactions with the environment. This allows α to predict how each neighbor will react to potential collisions and can reduce the uncertainty of the future state of the environment, enabling α to plan a safe and efficient path.

As the motion of the neighbors might be influenced by agents not visible to α, and other neighbors can appear at a later stage, α needs to continuously replan its path to account for this new information. Therefore, we compute a new plan at every timestep, where each plan is computed by simulating the execution of a sequence of actions for a time horizon of T timesteps, similar to the work in [Godoy *et al.*, 2014]. Algorithm 3 shows the pseudocode of the planning process using an action set, $Actions$, for the entire time horizon T.

Action Evaluation. To compare the actions available to α, each action is evaluated based on how much it helps the agent to progress towards its goal. We estimate such a progress by projecting the collision-free velocity computed as a result of

choosing this action onto the normalized goal-oriented vector of α:

$$\mathcal{R}_a = \mathbf{v}^{\text{new}} \cdot \frac{\mathbf{g} - \mathbf{p}}{\|\mathbf{g} - \mathbf{p}\|} \quad (4)$$

As a result, α promotes goal oriented behavior that is collision-free.

We consider two types of plans for α, with action spaces that allow different granularities of control for α's motion: a set of preferred velocities, and a set of navigation models. These two types of plans are complementary to each other and can be used to address a range of navigation problems. Planning in the space of different navigation models allows α to determine which high level behavior is best for interacting with neighbors in a given environment. This might be desired, for example, in human-robot interactions, where having a robot that exhibits human-like behavior allows for more natural interactions. Planning in the space of preferred velocities is recommended for traditional multi-robot navigation problems, where a fine-grained control of the robot's motion is typically required.

5.1 Planning in the space of preferred velocities

For a fine-grained motion control, we allow α to select among a set of 9 preferred velocities (Fig. 2). This set of velocities, defined in [Godoy *et al.*, 2014], is uniformly distributed in the space of possible directions of motions, enabling α to adapt its motion to different local conditions by attempting to move sideways or backwards from the goal when needed.

In the planning process, α simulates multiple plans of actions, where each plan involves the execution of a set of preferred velocities for T timesteps in the future. At each timestep, α updates its position along with the position of its neighbors, assuming that they try to reach their inferred goals using the inferred navigation models.

To determine which actions to include in a plan, we follow the approach in [Godoy *et al.*, 2014]. Figures 2a and 2b show an example of the plan generation process. We begin computing simple plans after taking the same action for each timestep (Fig. 2a), and progressively (if planning time allows) generate more complex plans composed by multiple actions (Fig. 2b). We compute the value of each simulated plan by

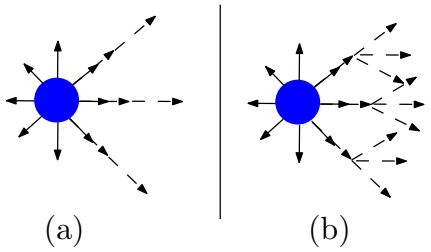

Figure 2: The 9 actions of *VelPlan* correspond to moving at 1.5 m/s with different angles with respect to the goal: $0°$, β, $-\beta$, $90°$, $-90°$, $180°$, $180° + \beta$, $180° - \beta$ and complete stop. We use $\beta = 45°$. (a) Simple plans consist of single actions for the entire time horizon. (b) More complex plans are computed if planning time allows.

Algorithm 3 Action Selection

```
for all a ∈ Actions do
    for t = 0, ..., T − 1 do
        simulate evolution of neighborhood dynamics
        R_a ← R_a + v^new · (g − p)/‖g − p‖
    end for
    R_a ← R_a / T
end for
a* ← arg max_{a∈Actions} R_a
```

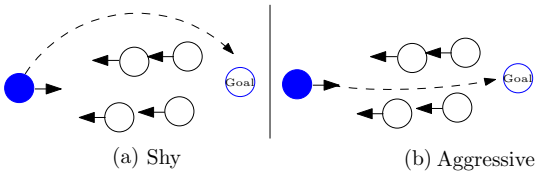

Figure 3: Example of ORCA behaviors. (a) Using a shy model, α decides to move around the group. (b) Using the aggressive model, instead, α makes way between the incoming agents to reach the goal faster.

computing $\mathcal{R}_a$ (Eq. 4) for each timestep in the time horizon. Finally, we average the reward value for all the plans that begin with the same initial action, and associate the averaged reward with the initial preferred velocity of the plan. The planning continues, generating more complex plans until a specified planning time limit is reached.

Using the space of preferred velocities, α has a more fine-grained control over its motion which allows it to adapt to a wide range of environments. We call this approach *VelPlan*.

5.2 Planning in the space of navigation models

Instead of making decisions in the space of preferred velocities, the agent could evaluate the space of navigation models.

Since the agent α has to anticipate collisions, we assume that, at each timestep, it can choose among any of the ORCA models presented in Section 3.1 (shy, aggressive, or tense). Depending on the selected model, α adapts its corresponding ORCA parameters, such as time horizon, sensing range, and assumption of reciprocity in avoiding collisions. Here α simulates each of the three models for the entire time horizon, computing at each timestep $\mathcal{R}_a$ (using Eq. 4). This type of planning is computationally more efficient than using preferred velocities (it requires only three simulations of T timesteps), and results in behaviors that can be explained from a social point of view. For example, α chooses a more conservative model (e.g. shy or tense) when congestion is predicted which allows the agent to avoid it, and a more aggressive model in scenarios with no obstacles, where α should move as fast as possible to its goal. Figure 3 shows different behaviors that α can assume using this type of planning. We call this planning method *SocialPlan*.

6 Experimental Setup and Results

We implemented our inference approach and our two planning methods, *VelPlan* and *SocialPlan*, in C++ and performed

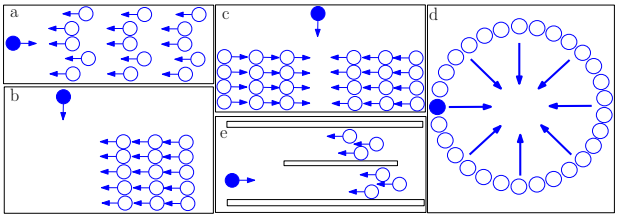

Figure 4: Scenarios. (a) Incoming. (b) Perp1. (c) Perp2. (d) Circle. (e) TwoPaths.

experiments across different simulation scenarios (see Figure 4). All simulations ran in real-time on a Core i7 CPU with 8 GB of memory. All reported results correspond to the average over 50 simulations. Our evaluation focuses on multi-robot navigation tasks, where we compare *VelPlan* to the ORCA framework and to a receding time-horizon approach which is similar to *VelPlan* but without an inference component (NoInference). We also evaluated our *SocialPlan* approach in a small social setting, and compared it with an ORCA-based fixed personality model.

In all scenarios, unless otherwise specified, we assigned the goal of each agent at the beginning of a simulation and we uniformly distributed all, but the unknown, navigation models (Section 3.1) over the set of agents A. To emulate agents that do not follow any model, we randomized the navigation parameters and goals of 5% of the agents after the initial goal and model allocation. We updated the position of α every $\Delta t = 0.1s$, which was also the planning time limit. We empirically set the values of the time horizon T to 50 timesteps, γ to 0.1%, and used a σ value of 10^{-4} (Eq. 1), as they provided the best performance compared to larger values.

Scenarios. We considered five scenarios (Figure 4): In the *Incoming* scenario, 30 agents move in the opposite direction of α, each having chosen a goal from 8 user-defined goals. In the *Perp1* scenario, a crowd of 100 agents moves in a perpendicular direction to α's motion, towards one goal. In the *Perp2* scenario, two groups of 50 agents each move in a perpendicular direction to α's motion, towards two opposite goals. In the *Circle* scenario, α and 63 agents are placed along the circumference of a circle and must reach their antipodal positions. Finally, in the *TwoPaths* scenario, six agents have to navigate through two narrow corridors towards six goals placed behind α's starting position. Here, α must choose between two homotopic paths based on the models of the agents perceived.

6.1 Model Prediction

We evaluate the accuracy of the Bayesian inference based on two criteria: *candidate* accuracy, where a prediction is correct if the true model/goal of the neighbor is in the set C_{mod} (candidate models/goals) and *predicted* accuracy, where a prediction is correct if the true model/goal is the one used for prediction of the neighbor's future motion.

Table 1 shows the accuracy of our *VelPlan* approach in all scenarios. In general, α can predict with high accuracy the models of its neighbors because of their distinctive ways of interacting with other agents. For example, shy and tense agents try to maintain a larger distance from other non-reactive or aggressive agents. The lowest accuracy is observed in the *TwoPaths* scenario. Here, the agents are constrained by the walls of the environment, which limits their motion and makes it more difficult to differentiate between similar models.

Accuracy	Incoming	Perp1	Perp2	Circle	TwoPaths
Candidate Model	94.4	95.5	95.1	94.5	86.4
Predicted Model	91.4	93.5	92.9	90.7	79.2
Candidate Goal	94.7	99.5	98.5	94.8	89.9
Predicted Goal	88.5	98.9	95.4	88.2	81.6

Table 1: Accuracy (%) of *VelPlan* in terms of goal and model prediction in our five scenarios.

Method	Incoming	Perp1	Perp2	Circle	TwoPaths
ORCA	51.22	83.28	38.08	5	11.7
NoInference	**0**	44.12	66.18	0.1	13.04
VelPlan	**0**	**6.96**	**28.74**	**0.04**	**0.02**

Table 2: Average collisions per iteration using ORCA, NoInference and *VelPlan*.

6.2 Number of collisions

Table 2 shows the average number of collisions between α and its neighbors, using *VelPlan*, ORCA and NoInference. Overall, using *VelPlan* results in significantly fewer collisions than the other two methods. However, even with *VelPlan*, α may collide with other agents. This is evident in crowded environments, such as $Perp2$, where the number of agents limits the capacity of α to find a safe motion, for example, when facing non-reactive agents coming in opposite directions. Using either NoInference or ORCA, α runs into more collisions because α incorrectly assumes that agents will share the effort to avoid collisions, which is not always true.

6.3 Time-efficiency

To measure the time-efficiency of the path that α takes, we measured the *interaction overhead*, i.e. the time that it takes for the agent to resolve interactions with the other agents and obstacles. The overhead is zero if α is able to move at full speed in a straight line to its goal (i.e. without deviating from its goal-oriented path). Figure 5 compares the interaction overhead in all our scenarios using different simulation methods. As can be seen from the figure, predicting the behavior

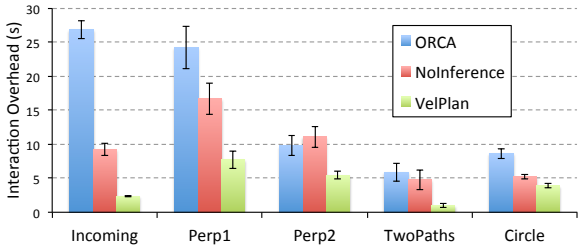

Figure 5: Interaction overhead for all evaluated approaches. In all scenarios, *VelPlan* outperforms ORCA and NoInference. The error bars denote the standard error of the mean.

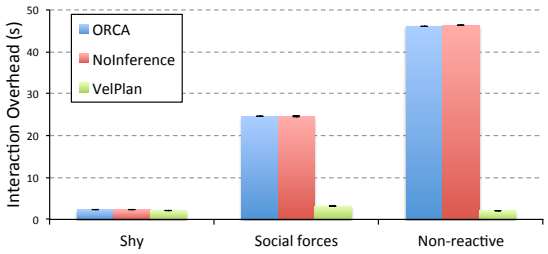

Figure 6: Interaction overhead for agents using the same model in the *Incoming* scenario. In all cases, *VelPlan* matches or outperforms ORCA and NoInference.

of other agents clearly improves the time-efficiency of α's navigation. We can observe that *VelPlan* significantly outperforms both NoInference and ORCA in all scenarios. The performance difference is the largest in the $Incoming$ scenario, where α has enough room to maneuver to avoid non-reactive agents, while also being able to move close to agents that will defer to its motion, such as the shy agents. It is worth noting the difference in performance between *Perp1* and *Perp2*. The latter scenario is particularly complex, as previously shown with the lower accuracy values and the large number of collisions. These results, together with Table 2, demonstrate the utility of the model and goal estimation in the computation of a safe and efficient path for α.

Single model. We also evaluated the interaction overhead of α using *VelPlan*, NoInference and ORCA in the *Incoming* scenario, when all other agents use either the shy model, the social force model, or are non-reactive. Results shown in Figure 6 indicate that, against shy agents, all approaches are able to find paths with low interaction overhead. Here, the shy agents make space for α, hence the deviation from the goal path is minimal. When the other agents are using social forces, however, only *VelPlan* realizes that it is best to move around the group in order to reach the goal faster. In contrast, using ORCA or NoInference, α gets trapped and pushed back by these agents because it incorrectly assumes that the other agents will anticipate potential collisions. Similar behavior occurs with non-reactive agents, where only *VelPlan* is able to successfully avoid getting stuck.

6.4 Social Planning

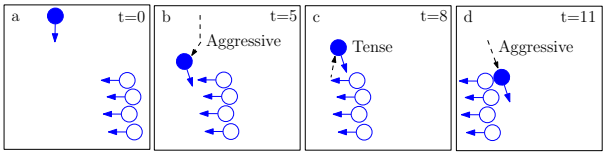

Figure 7: Social Planning example. (a) Initial positions (b) The agent uses aggressive behavior until its motion is constrained by the group. (c) The agent switches to tense behavior which pushes it away from the group. (d) After the group has passed by, the agent assumes again an aggressive behavior to move fast and unconstrained to its goal.

To analyze the robustness of our planning approach to different action spaces, we evaluated the behavior and the interaction overhead of our *SocialPlan* approach in a small scenario (see Figure 7a), where α crosses path with 4 other agents that employ social forces. Here, using *SocialPlan* means that α had to choose between being shy, aggressive or tense. We observe that, using our approach, α chooses an aggressive personality in the beginning as it gets close to the group (Fig. 7b). Once it realizes it will be pushed away from its goal, α switches to a tense personality (Fig. 7c), which allows it to move back for a while and let the group pass. Once this happens, it adopts an aggressive personality which enables α to quickly move to its goal (Fig. 7d).

	Neighbors		
Agent α	Shy	Social Forces	Mixed
Shy	2	3.7	2.6
Aggressive	**0**	31.5	2.7
Tense	36.8	27.6	4.94
SocialPlan	**0**	**1.5**	**0.9**

Table 3: Interaction overhead(s) in Social Planning for the scenario in Figure 7a.

We also compared the interaction overhead of α using *SocialPlan* versus using a fixed personality model (shy, tense or aggressive) during the entire simulation. We varied the model used by the other 4 agents (only shy, only social forces or a mix of agent models). Table 3 shows the corresponding results. We can see that, due to the ability of *SocialPlan* to determine the types of the other agents and to switch between behaviors in real time, α finds velocities that increase its time-efficiency as compared to choosing a fixed personality model.

7 Conclusions and Future Work

In this paper, we introduced a novel technique to improve the safety and time-efficiency of an agent navigating in an unknown environment. Our approach determines the navigation model and goal of other agents in the environment, and uses these estimates to predict their future states.

We have assumed a fix number of representative navigation models are given to us. This is a reasonable assumption, given that most existing multi-robot navigation approaches generate behaviors that can be broadly classified as reactive, non-responsive, or anticipatory. However, our approach does not attempt to learn navigation models outside of the specified set of models. To address this limitation, we would like to explore methods to learn new navigation models from observations of the agents' motions. Another possible line for future work is to combine our local approach with an adaptive global planning scheme. The work in [Jansen and Sturtevant, 2008; Golledge, 1995; Van Toll *et al.*, 2012] can provide some interesting ideas in this direction.

Acknowledgments: Partial support for this work is acknowledged from the University of Minnesota Informatics Institute and NSF through grants #CNS-1544887 and #CHS-1526693.

References

[Barrett and Stone, 2015] Samuel Barrett and Peter Stone. Cooperating with unknown teammates in complex do-

mains: A robot soccer case study of ad hoc teamwork. In *Proc. AAAI Conference on Artificial Intelligence*, 2015.

[Barrett *et al.*, 2013] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld. Teamwork with limited knowledge of teammates. In *Proc. AAAI Conference on Artificial Intelligence*, 2013.

[Choi *et al.*, 2014] Sungjoon Choi, Eunwoo Kim, and Songhwai Oh. Real-time navigation in crowded dynamic environments using gaussian process motion control. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3221–3226. IEEE, 2014.

[Cirillo *et al.*, 2014] Marcello Cirillo, Tansel Uras, and Sven Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 232–239. IEEE, 2014.

[Fiorini and Shiller, 1998] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using Velocity Obstacles. *Int. J. Robotics Research*, 17:760–772, 1998.

[Godoy *et al.*, 2014] Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. Anytime navigation with progressive hindsight optimization. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.

[Godoy *et al.*, 2016] Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. Implicit coordination in crowded multi-agent navigation. In *Proc. AAAI Conference on Artificial Intelligence*, 2016.

[Golledge, 1995] Reginald G Golledge. *Path selection and route preference in human navigation: A progress report*. Springer, 1995.

[Guy *et al.*, 2011] Stephen J Guy, Sujeong Kim, Ming C Lin, and Dinesh Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–52. ACM, 2011.

[Helbing *et al.*, 2000] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[Jansen and Sturtevant, 2008] M.R. Jansen and N.R. Sturtevant. Direction maps for cooperative pathfinding. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 185–190, 2008.

[Karamouzas *et al.*, 2014] Ioannis Karamouzas, Brian Skinner, and Stephen J Guy. Universal power law governing pedestrian interactions. *Physical review letters*, 113(23), 2014.

[Khatib, 1986] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robotics Research*, 5(1):90–98, 1986.

[Kim *et al.*, 2014] Sujeong Kim, Stephen J Guy, Wenxi Liu, David Wilkie, Rynson WH Lau, Ming C Lin, and Dinesh Manocha. BRVO: Predicting pedestrian trajectories using velocity-space reasoning. *The Int. J. of Robotics Research*, 34(2):201–217, 2014.

[Luna and Bekris, 2011] Ryan Luna and Kostas E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *International Joint Conference on Artificial Intelligence*, pages 294–300, 2011.

[Martinez-Gil *et al.*, 2015] Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. Strategies for simulating pedestrian navigation with multiple reinforcement learning agents. *Autonomous Agents and Multi-Agent Systems*, 29(1):98–130, 2015.

[Melo and Veloso, 2011] Francisco S Melo and Manuela Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

[Pelechano *et al.*, 2007] N. Pelechano, J.M. Allbeck, and N.I. Badler. Controlling individual agents in high-density crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 99–108, 2007.

[Sharon *et al.*, 2015] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[Solovey *et al.*, 2015] Kiril Solovey, Jingjin Yu, Or Zamir, and Dan Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems*, 2015.

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

[van den Berg *et al.*, 2011] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Proc. International Symposium of Robotics Research*, pages 3–19. Springer, 2011.

[Van Toll *et al.*, 2012] Wouter G Van Toll, Atlas F Cook, and Roland Geraerts. Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds*, 23(1):59–69, 2012.