Autonomous Navigation in Dynamic Social Environments using Multi-Policy Decision Making

Dhanvin Mehta[1], Gonzalo Ferrer[1] and Edwin Olson[1]

***Abstract*— In dynamic environments crowded with people, robot motion planning becomes difficult due to the complex and tightly-coupled interactions between agents. Trajectory planning methods, supported by models of typical human behavior and personal space, often produce reasonable behavior. However, they do not account for the future closed-loop interactions of other agents with the trajectory being constructed. As a consequence, the trajectories are unable to anticipate cooperative interactions (such as a human yielding), or adverse interactions (such as the robot blocking the way).**

In this paper, we propose a new method for navigation amongst pedestrians in which the trajectory of the robot is not explicitly planned, but instead, a planning process selects one of a set of closed-loop behaviors whose utility can be predicted through forward simulation. In particular, we extend Multi-Policy Decision Making (MPDM) [1] to this domain using the closed-loop behaviors *Go-Solo*, *Follow-other*, and *Stop*. By dynamically switching between these policies, we show that we can improve the performance of the robot as measured by utility functions that reward task completion and penalize inconvenience to other agents. Our evaluation includes extensive results in simulation and real-world experiments.

I. Introduction

Maneuvering in dynamic social environments is challenging due to uncertainty associated with estimating and predicting future scenarios arising from the complex and tightly-coupled interactions between people. Sensor noise, action execution uncertainty, tracking data association errors, etc. make this problem harder.

Trajectory planning methods such as [2], [3] use models of human behavior to propagate the state of the environment, but may fail to account for the closed-loop coupled interactions of agents.

A robot needs to exhibit a wide range of emergent behaviors to successfully deal with the various situations that are likely to arise in social environments. For instance, navigating in a hallway with freely moving people is different than a situation where people crowd around a door to exit a room. Several navigation algorithms [2]–[16] that calculate a single navigation solution may find it hard to deal with all these scenarios. This inflexibility may result in undesirable solutions under challenging configurations.

In this work, we propose a novel approach to motion planning amongst people. Instead of computing a trajectory directly or relying on a single algorithm, we evaluate a set of closed-loop policies by predicting their utilities through forward simulation that captures the coupled interactions between the agents in the environment.

[1]The authors are associated with the University of Michigan, Ann Arbor. {dhanvinm,gferrerm,ebolson}@umich.edu.
This work was supported by DARPA D13AP00059.

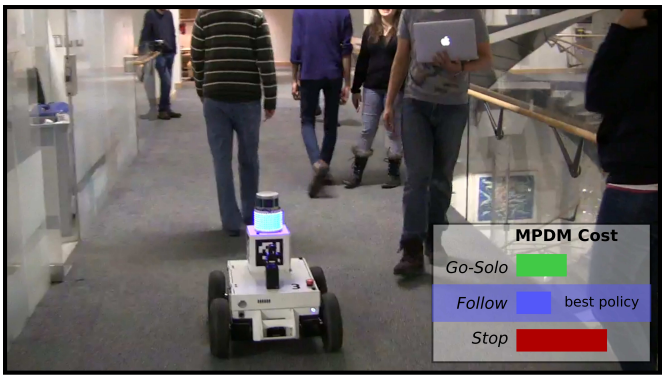


Fig. 1. Our approach implemented and tested using the MAGIC [17] robot platform. We show that our algorithm is able to navigate successfully on an indoor environment amongst people. MPDM allows the robot to choose between policies. In this case, the robot decides to *Follow* the person in front rather than try to overtake him.

We extend MPDM [1] to navigate in dynamic, unstructured environments where the dynamic agents (humans) can instantaneously stop or change direction without signaling. To achieve this, we use different and complementary policies than those considered by Cunningham *et al.* [1]: *Go-Solo*, *Follow-other* and *Stop*. In order for the robot's emergent behavior to be socially acceptable, each policy's utility is estimated trading-off the distance traveled towards the goal (*Progress*) with the potential disturbance caused to fellow agents (*Force*).

Dynamically switching between the candidate policies allows the robot to adapt to different situations. For instance, the best policy might be to *Stop* if the robot's estimation uncertainty is large. Similarly, the robot may choose to *Follow* a person through a cluttered environment. This may make the robot slower, but allows it to get a clearer path since humans typically move more effectively in crowds, as depicted in Fig. 1.

Due to the low computational requirements of evaluating our proposed set of policies, the robot can re-plan frequently, which helps reduce the impact of uncertainty. We show the benefits of switching between multiple policies in terms of navigation performance, quantified by metrics for progress made and inconvenience to fellow agents. We demonstrate the robustness of MPDM to measurement uncertainty and study the effect of the conservatism of the state estimator through simulation experiments (Sec. VI). Finally, we test the MPDM on a real environment and evaluate the results (Sec. VII).

II. Related Work

In a simulated environment, van den Berg *et al.* [18] proposed a multi-agent navigation technique using *velocity obstacles* that guarantees a collision-free solution assuming a fully-observable world. From the computer graphics community, Guy *et al.* [19] extended this work using *finite-time velocity obstacles* to provide a locally collision-free solution that was less conservative as compared to [18]. However, the main drawback of these methods is that they are sensitive to imperfect state estimates and make strong assumptions that may not hold in the real world.

Several approaches attempt to navigate in social environments by traversing a Potential Field (PF) [20] generated by a set of pedestrians [4]–[6]. Huang *et al.* [9] used visual information to build a PF to navigate. In the field of neuroscience, Helbing and Molnár [21] proposed the Social Force Model, a kind of PF approach that describes the interactions between pedestrians in motion.

Unfortunately, PF approaches have some limitations, such as local minima or oscillation under certain configurations [22]. These limitations can be overcome to a certain degree by using a global information plan to avoid local minima [23]. We use this same idea in our method by assuming that a global planner provides reachable goals, i.e., there is a straight line connection to those positions ensuring feasibility in the absence of other agents.

Inverse Reinforcement Learning-based approaches [10]–[13] can provide good solutions by predicting social environments and planning through them. However, their effectiveness is limited by the training scenarios considered which might not be a representative set of the diverse situations that may arise in the real world.

An alternative approach looks for a pedestrian leader to follow, thus delegating the responsibility of finding a path to the leader, such as the works of [7], [14], [15]. In this work, *Follow* becomes one of the policies that the robot can choose to execute as an alternate policy to navigating.

Some approaches [2], [8], [16] plan over the predicted trajectories of other agents. However predicting the behavior of pedestrians is challenging and the underlying planner must be robust to prediction errors.

POMDPs provide a principled approach to deal with uncertainty, but they quickly become intractable. Foka *et al.* [3] used POMDPs for robot navigation in museums. Cunningham *et al.* [1] show that, by introducing a number of approximations (in particular, constraining the policy to be one of a finite set of known policies), that the POMDP can be solved using MPDM. In their original paper, they use a small set of lane-changing policies; in this work, we explore an indoor setting in which the number and complexity of candidate policies is much higher.

III. Problem formulation

Our model of the environment consists of static obstacles (e.g. walls or doors) and a set of freely moving dynamic agents, assumed to be people.

The robot maintains *estimates* of the states of observable agents. The state $\boldsymbol{x}_i \in \mathcal{X}_i$ for agent i (including the robot) consists of its position $\boldsymbol{p}_i$, velocity $\boldsymbol{v}_i$ and a goal point $\boldsymbol{g}_i$.

$$\boldsymbol{x}_i = [\, \boldsymbol{p}_i, \boldsymbol{v}_i, \boldsymbol{g}_i]^\top, \tag{1}$$

where each of $\boldsymbol{p}_i$, $\boldsymbol{v}_i$ and $\boldsymbol{g}_i$ are two-dimensional vectors. The motion of agents is modeled according to a simple dynamics model in which acceleration, integrated over time, results in a velocity. The force, and hence the acceleration, is computed using a potential field method that incorporates the effects of obstacles and a goal point.

Let N be the number of agents including the robot. The joint state space of the system is $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_N$. The collective state $\boldsymbol{x}(t) \in \mathcal{X}$ includes the robot state plus all the agents visible to the robot at time t.

Our observation model $P(\boldsymbol{z}|\boldsymbol{x})$ is assumed to be Markovian, where the joint observations $\boldsymbol{z}$ are the pedestrians' positions. In Sec. V we will discuss the impact that the estimator's posterior distribution $P(\boldsymbol{x}|\boldsymbol{z})$ has on our approach. For each pedestrian, the goal $\boldsymbol{g}_i$ is not directly observable through $\boldsymbol{z}$. It is assumed to be one of a small set of salient points and is estimated using a naive Bayes Classifier. For the robot, the goal $\boldsymbol{g}_r$ is provided by a higher level planner.

The agent dynamics are defined by the following differential constraints:

$$\dot{\boldsymbol{x}}_i = [\boldsymbol{v}_i, \boldsymbol{a}_i, \boldsymbol{0}]^\top, \tag{2}$$

The action $\boldsymbol{a}_i \in \mathcal{A}_i$ corresponds to the acceleration governing the system dynamics and is determined by the policy ξ_i followed by the agent (Sec. IV).

The transition function maps a given state $\boldsymbol{x}_i$ and an action $\boldsymbol{a}_i$ to a new state $T : \mathcal{X}_i \times \mathcal{A}_i \mapsto \mathcal{X}_i$. Thus, the corresponding transition equation is expressed as

$$T(\boldsymbol{x}_i, \boldsymbol{a}_i) = \boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \int_t^{t+\Delta t} \dot{\boldsymbol{x}}_i(\tau, \boldsymbol{a}_i) d\tau. \tag{3}$$

The system is constrained to a maximum velocity $|v|_{max}$ for each agent.

IV. Navigation policies

We approach this problem by reasoning over a discrete set of high-level closed-loop policies.

$$\boldsymbol{\xi} = \{\textit{Go-Solo}, \textit{Follow}_j, \textit{Stop}\}, \tag{4}$$

where $\textit{Follow}_j$ refers to the policy of following agent j. A robot in an environment with 10 observable agents has a total of 12 candidate policies, much greater than the 3 policies considered by Cunningham *et al.* [1].

Each policy maps a joint state of the system to an action via a potential field $\xi_i \in \boldsymbol{\xi} : \mathcal{X} \mapsto \mathcal{A}_i$.

A. Go-Solo Policy

An agent executing the *Go-Solo* policy treats all other agents as obstacles and uses a potential field based on the Social Force Model (SFM) [6], [21] to guide it towards its goal.

Let $\boldsymbol{e}_{\boldsymbol{p}_i \to \boldsymbol{g}_i}$ be the unit vector towards the goal from the agent i. The attractive force acting on the agent is given by:

$$\boldsymbol{f}_i^{attr}(\boldsymbol{x}) = k_{gs}\boldsymbol{e}_{i \to \boldsymbol{g}_i}. \tag{5}$$

We model the interactions with other agents in the scene based on the SFM :

$$\boldsymbol{f}_{i,j}^{int}(\boldsymbol{x}) = a_p e^{-d_{i,j}/b_p} \cdot \boldsymbol{e}_{j \to i}, \tag{6}$$

where $\{a_p, b_p\}$ are the SFM parameters for people, $\boldsymbol{e}_{j \to i}$ is the unit vector from j to i and $d_{i,j}$ is the distance between them scaled by an anisotropic factor as in [6] .

Similarly, each obstacle $o \in O$ in the neighborhood of the agent exerts a repulsive force $\boldsymbol{f}_{i,o}^{obs}(\boldsymbol{x})$ on agent i according to different SFM parameters $\{a_o, b_o\}$,

$$\boldsymbol{f}_{i,o}^{obs}(\boldsymbol{x}) = a_o e^{-d_{i,o}/b_o} \cdot \boldsymbol{e}_{o \to i}. \tag{7}$$

The resultant force is a summation of all the forces described above:

$$\boldsymbol{f}_i(\boldsymbol{x}) = \boldsymbol{f}_i^{attr}(\boldsymbol{x}) + \sum_{j \neq i} \boldsymbol{f}_{i,j}^{int} + \sum_{o \in O} \boldsymbol{f}_{i,o}^{obs} \tag{8}$$

The action governing the system propagation (2) is calculated as $\boldsymbol{a}_i = \boldsymbol{f}_i$ (without loss of generality, we assume unit mass). We assume that all agents besides the robot always use this *Go-Solo* policy.

B. Follow Policy

In addition to the *Go-Solo* policy, the robot can use the *Follow* policy to deal with certain situations. Our intuition is that in a crowd, the robot may choose to *Follow* another person sacrificing speed but delegating the task of finding a path to a human. *Following* could also be more suitable than overtaking a person in a cluttered scenario as it allows the robot to *Progress* towards its goal without disturbing other agents (low *Force*). We propose a reactive *Follow* policy, making minor modifications to the *Go-Solo* policy.

According to the *Follow* policy, the robot r *chooses* to follow another agent, the leader, denoted by l. We can apply the same procedure explained in Sec. IV-A with the modification that the robot is attracted to the leader rather than the goal. Let $\boldsymbol{e}_{\boldsymbol{p}_r \to \boldsymbol{p}_l}$ be the unit vector from the robot's position to the leader's position. The attractive force

$$\boldsymbol{f}_r^{attr}(\boldsymbol{x}) = k_f \boldsymbol{e}_{\boldsymbol{p}_r \to \boldsymbol{p}_l}, \tag{9}$$

steers the robot trajectory towards the leader. The other agents and obstacles continue to repel the robot as described in (8).

C. Stop Policy

The last of the policies available to the robot is the *Stop* policy, where the robot decelerates until it comes to a complete stop, according to the following force

$$\boldsymbol{f}_r(\boldsymbol{x}) = -f_{max}\boldsymbol{e}_{\boldsymbol{v}_r}, \tag{10}$$

where $\boldsymbol{e}_{\boldsymbol{v}_r}$ is the unit vector in the direction of the robot's velocity.

In the following section, we will describe the procedure to choose between these three policies, and the resultant force $\boldsymbol{f}_i(x, \xi)$ will be expressed as a function of the policy ξ.

Algorithm 1 MPDM($\boldsymbol{x}, z, t_H, N_s$)

for $\xi \in \boldsymbol{\xi}$ **do**
  for $s = 1 \dots N_s$ **do**
    $\boldsymbol{x}_s \sim P(\boldsymbol{x}|\boldsymbol{z})$ // Sampling over the posterior.
    $C(\boldsymbol{x}_s, \xi) = \text{simulate\_forward}(\boldsymbol{x}_s, \xi, t_H)$
  end for
end for
return $\xi^* = \arg\min_\xi (E_{\boldsymbol{x}}\{C(\boldsymbol{x}, \xi)\})$

Algorithm 2 simulate_forward($\boldsymbol{x}, \xi, t_H$)

$\hat{\boldsymbol{X}} = \{\}$
for $t' = t, t + \Delta t, \dots, t_H$ **do**
  $\dot{\hat{\boldsymbol{x}}}_r(t') = \boldsymbol{f}_r(\boldsymbol{x}(t'), \xi)$ // Propagate robot
  $\hat{\boldsymbol{x}}_r(t' + \Delta t) = \hat{\boldsymbol{x}}_r(t') + \int_{\Delta t} \dot{\hat{\boldsymbol{x}}}_r(\tau) d\tau$
  for $i \in 1 \dots r-1, r+1 \dots N$ **do**
    $\dot{\hat{\boldsymbol{x}}}_i(t') = \boldsymbol{f}_i(\boldsymbol{x}(t'), \textit{Go-Solo})$ // Propagate people
    $\hat{\boldsymbol{x}}_i(t' + \Delta t) = \hat{\boldsymbol{x}}_i(t') + \int_{\Delta t} \dot{\hat{\boldsymbol{x}}}_i(\tau) d\tau$
  end for
  $\hat{\boldsymbol{x}}(t' + \Delta t) = \{\boldsymbol{x}_1(t' + \Delta t), \dots, \boldsymbol{x}_N(t' + \Delta t)\}$
  $\hat{\boldsymbol{X}}.\textit{append}(\hat{\boldsymbol{x}}(t' + \Delta t))$
end for
$C = -\alpha PG(\hat{\boldsymbol{X}}) + F(\hat{\boldsymbol{X}})$ // Calculate cost
return C

V. Multi-Policy Decision Making

Decision making is constantly recalculated in a Receding Horizon fashion. MPDM chooses the policy $\xi \in \boldsymbol{\xi}$ that optimizes the following objective function (Alg. 1):

$$\xi^* = \arg\min_{\xi} E_{\boldsymbol{x}}\{C(\boldsymbol{x}, \xi)\}. \tag{11}$$

The cost $C(\boldsymbol{x}, \xi)$ is associated with the current joint state $\boldsymbol{x}$ upon choosing policy ξ. In order to obtain a cost function, for each agent, we predict a trajectory $\hat{\boldsymbol{X}}_i(\xi)$ governed by the particular policy ξ executed by the agent:

$$\hat{\boldsymbol{X}}_i(\xi) = \{\hat{\boldsymbol{x}}_i(t + \Delta t, \xi), \dots, \hat{\boldsymbol{x}}_i(t_H - \Delta t, \xi), \hat{\boldsymbol{x}}_i(t_H, \xi)\}. \tag{12}$$

We forward simulate the joint state $\boldsymbol{x}$ until a time horizon t_H by applying (3) iteratively and simultaneously for each agent, as can be seen in Alg. 2. We obtain a propagation of the agents' trajectories as well as the robot's $\hat{\boldsymbol{X}}(\xi)$, which are especially interesting since the trajectories react to their interactions with the robot's proposed plan, and vice-versa.

A. The Cost Function

Our cost function consists of two different components: *Force* which captures the potential disturbance that the robot causes in the environment and *Progress* which indicates progress made towards the goal.

Force: We use the maximum repulsive force (6) exerted on another agent (except the leader, if any) as a proxy for the potential disturbance caused to the environment by the robot.

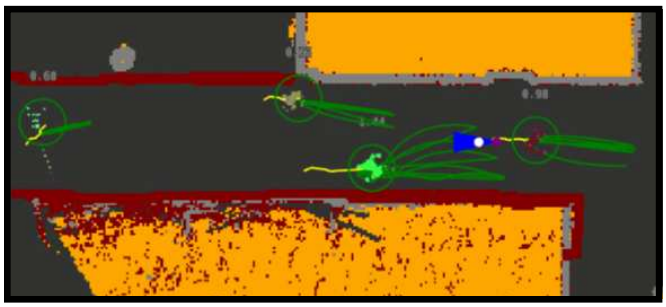


Fig. 2. The graphical interface during a real-world experiment. We use a grid-map encoding static obstacles as red/gray cells, free spaces as black cells, and unknown regions as yellow cells. The robot (blue triangle) uses 10 samples to estimate future trajectories (green lines) for tracked agents (green circles), and calculates expected scores based on these samples. The yellow lines denote the past trajectories of the tracked agents.

$$F(\hat{\boldsymbol{X}}(\xi)) = \begin{cases} \sum_{t'=t}^{t_H} \max_{j \neq i} \|\boldsymbol{f}_{r,j}^{int}(\hat{\boldsymbol{x}}(t', \xi))\| & \xi_r \neq \textit{Follow} \\ \sum_{t'=t}^{t_H} \max_{j \neq i, l} \|\boldsymbol{f}_{r,j}^{int}(\hat{\boldsymbol{x}}(t', \xi))\| & \xi_r = \textit{Follow} \end{cases} \tag{13}$$

Progress: We encourage the robot for the distance-made-good during the planning horizon.

$$PG(\hat{\boldsymbol{X}}(\xi)) = \left(\hat{\boldsymbol{x}}_i(t_H, \xi) - \boldsymbol{x}_r(t)\right) \cdot \boldsymbol{e}_{r \to \boldsymbol{g}_r}, \tag{14}$$

where $\boldsymbol{e}_{r \to \boldsymbol{g}_r}$ is the unit vector from the current position of r to the goal $\boldsymbol{g}_r$.

The resultant cost function is a linear combination of both

$$C(\boldsymbol{x}, \xi) = -\alpha PG(\boldsymbol{x}, \xi) + F(\boldsymbol{x}, \xi), \tag{15}$$

where alpha is a weighting factor.

Obtaining a closed form of the cost expectation, as expressed in (11), is believed to be impossible. For this reason, we use a sampling technique to approximate the expected cost:

$$E_{\boldsymbol{x}}\{C(\boldsymbol{x}, \xi)\} \sim \sum_{s \in S} w_s C(\boldsymbol{x}_s, \xi), \tag{16}$$

where S is the set of samples drawn from the distribution $P(\boldsymbol{x}|\boldsymbol{z})$. These samples seed the forward propagation of the joint state, resulting in a set of different future trajectories. Thus, the robot's behavior reflects not only the mean state estimates of the other agents, but also the uncertainty associated with those estimates.

Estimation uncertainty and measurement noise affect the quality of sampled future trajectories and thereby system performance. In Sec. VI we will show that the flexibility of choosing between multiple policies makes our approach robust to measurement noise, and err on the side of caution. Fig. 2 shows a scenario where the robot (in blue) chooses to follow the agent ahead of it after predicting multiple trajectories of visible agents for each candidate policy and approximating the expected cost associated with each policy.

VI. Simulations

We simulate two indoor domains, freely traversed by a set of agents while the robot tries to reach a goal. One simulation

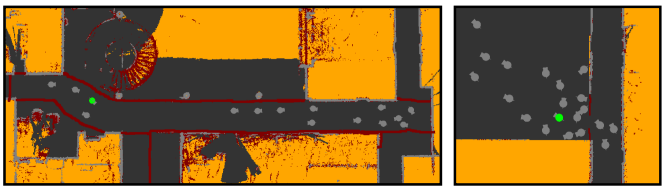


Fig. 3. The simulated indoor domains chosen to study our approach. *Left*: The hallway domain where 15 agents are let loose with the robot and they patrol the hallway while the robot tries to reach its destination. *Right*: The doorway domain where 15 agents whose goal is reaching the bottom right of the map through the door. These two domains present the robot with a set of diverse, but realistic indoor situations (crossing agents in a hallway, queuing and dense crowding near a doorway).

'epoch' consists of a random initialization of agent states followed by a 5 minute simulated run at a granularity $\Delta t = 0.1$s. The number of samples used to approximate the expected cost according to (16) $N_s = 50$. We use the Intel i7 processor and 8GB RAM for our simulator and LCM [24] for inter-process communication.

Every 333ms (policy election cycle), MPDM chooses a policy ξ. Although the policy election is slow, the robot is responsive as the policies themselves run at over 100Hz.

We assume that the position of the robot, agents, the goal point, and obstacles are known in some local coordinate system. However, the accuracy of motion predictions is improved by knowing more about the structure of the building since the position of walls and obstacles can influence the behavior of other agents over the 3 second planning horizon. Our implementation achieves these through a global localization system with a known map, but our approach could be applied more generally.

A. Domains

The hallway domain (Fig. 3-*Left*) is modeled on a $3m \times 25m$ hallway at the University of Michigan.

The doorway domain (Fig. 3-*Right*) consists of a room with a door at the corner of the room leading into a hallway. The robot and all agents try to reach the hallway through the door.

Based on the observed empirical distributions over some runs (Fig. 4-*Left*), we set $\alpha = 15$ so that *Force* and *Progress* have similar impact on the cost function.

The maximum permitted acceleration is $3m/s^2$ while the maximum speed $|v|_{max}$ is set to $0.8m/s$. MPDM is carried out at 3Hz to match the frequency of the sensing pipeline for state estimation in the real-world experiment. The planning horizon is $3s$ into the future.

B. Evaluation Metrics

Evaluating navigation behavior objectively is a challenging task and unfortunately, there are no standard metrics. We propose three metrics that quantify different aspects of the emergent navigation behavior of the robot.

1) *Progress* (PG) - measures distance made good, as presented in (14).
2) *Force* (F) - penalizes close encounters with other agents, calculated at each time step according to (13).

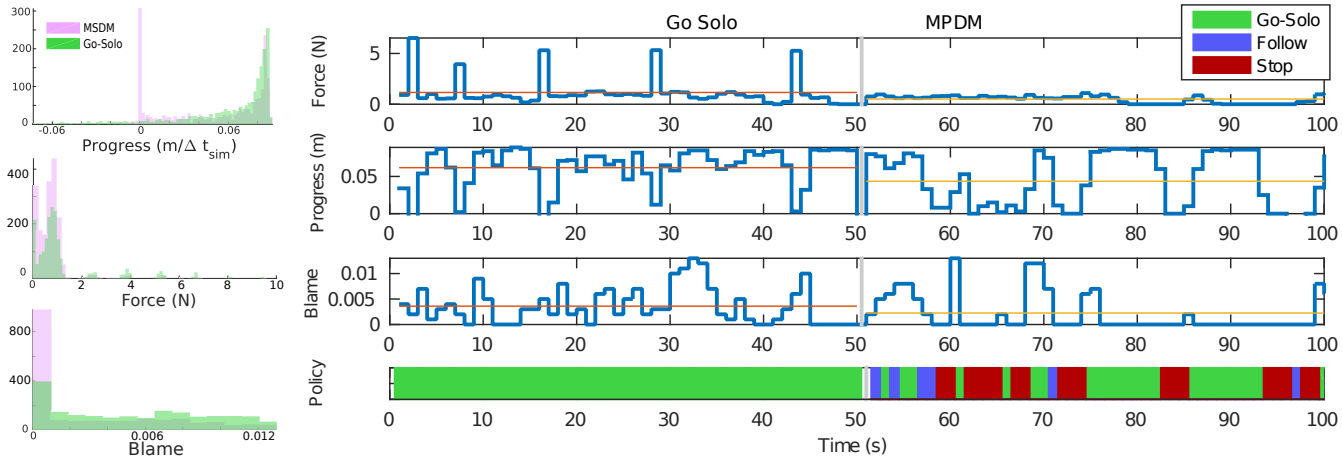


Fig. 4. Qualitative evaluation of some simulation runs comparing MPDM and the exclusive use of *Go-Solo*. *Right*: Temporal evolution in the hallway domain where first the robot ran a fixed *Go-Solo* policy for 50s followed by MPDM for the next 50s. The horizontal red lines indicate the average values for the trajectory. The *Go-Solo* performance makes a lot of *Progress* but incurs high *Force* and *Blame*, manifesting as undesired peaks. In the next 50s, the MPDM *Force* curve is almost flat, meaning that nearby interactions are reduced drastically and *Blame* is reduced significantly. *Left*: Distributions of the evaluation metrics - *Force*, *Blame* and *Progress* respectively over 3k seconds. Ideal behavior would give rise to high *Progress* and low *Force*. The higher valued modes for *Force* denote undesirable behavior (close encounters), which MPDM is able to avoid.

3) *Blame* (B) - penalizes velocity at the time of close encounters which is not captured by *Force*. Let $\boldsymbol{p}^*_{ij}(t)$ be the point on the line segment joining $\boldsymbol{p}_i(t) + \boldsymbol{v}_i\tau$ and $\boldsymbol{p}_i(t)$ that is closest to $\boldsymbol{p}_j(t)$. Then,

$$B_i(t) = \max_j \Phi(||\boldsymbol{p}^*_{ij}(t) - \boldsymbol{p}_j(t)||), \tag{17}$$

where Φ is a sigmoid function and τ is set to $0.5s$ in our experiments. We further motivate this metric in Sec. VI-C.

C. Empirical Validation

To empirically validate our algorithm, we run the MPDM on both domains, assuming perfect observations. Fig. 4-*Right* shows the performance of the MPDM as compared to using the *Go-Solo* policy exclusively. During the initial 50s (*Go-Solo*) the robot makes a lot of *Progress* but incurs high *Force* and *Blame* due to undesired motion, aggressively forcing its way forward even when it is very close to another agent and hindering its path. For the next 50s, the MPDM dynamically switches policies maintaining low *Force* and *Blame* no longer inconveniencing other agents.

This observation is strengthened by the empirical distributions of the metrics generated from 30k samples. We notice that the *Force* and *Blame* distributions have greater density at lower values for MPDM. Negative *Progress*, which occurs when the agents come dangerously close to each other exerting a very strong repulsive force, is absent in MPDM as the agent would rather stop.

As stated before, PF are subject to local minima problems, so our simulation environment is susceptible to frontal crossings resulting in temporal "freezing behaviors" [16], where both agents remain for some seconds unable to escape this situation. This behavior is only temporal due to the dynamic nature of the environment, but it is atypical of human beings. This limitation motivates the introduction of *Blame* as a velocity sensitive metric to better analyze navigation behavior in our simulator. In real environments, *Blame* and *Force* are strongly correlated since people are not likely to collide into a stationary robot.

D. Experiments with Noise

MPDM is a general approach in the sense that it makes no assumptions about the quality of state estimation. The more accurate our model of the dynamic agents, the better is the accuracy of the predicted joint states. Most models of human motion, especially in complicated situations, fail to predict human behavior accurately. This motivates us to extensively test how robust our approach is to noisy environments.

In our simulator, the observations $\boldsymbol{z}$ are modeled using a stationary Gaussian distribution with uncorrelated variables for position, speed and orientation for the agent. We parameterize this uncertainty by a scale factor k_z: $\{\sigma_{p_x}, \sigma_{p_y}, \sigma_{|v|}, \sigma_\theta\} = k_z \times \{2cm, 2cm, 2cm/s, 3°\}$. The corresponding diagonal covariance matrix is denoted by $\text{diag}(\sigma_{p_x}, \sigma_{p_y}, \sigma_{|v|}, \sigma_\theta)$. We do not perturb the goal. These uncertainties are propagated in the posterior estate estimation $P(\boldsymbol{x}|\boldsymbol{z})$.

The robot's estimator makes assumptions about the observation noise which may or may not match the noise injected by the simulator. This can lead to over and under-confidence which affects decision making. In this section, we explore the robustness of the system in the presence of these types of errors. We define the assumed uncertainty by the estimator through a scale factor k_e, exactly as described above.

For each of the domains considered, we evaluate the performance of the system by:

- varying k_z for a fixed k_e to understand how MPDM performs when varying uncertainty in the environment.
- varying $\frac{k_e}{k_z}$ to understand how MPDM performs when the robot's estimator overestimates/underestimates the uncertainty in the environment.

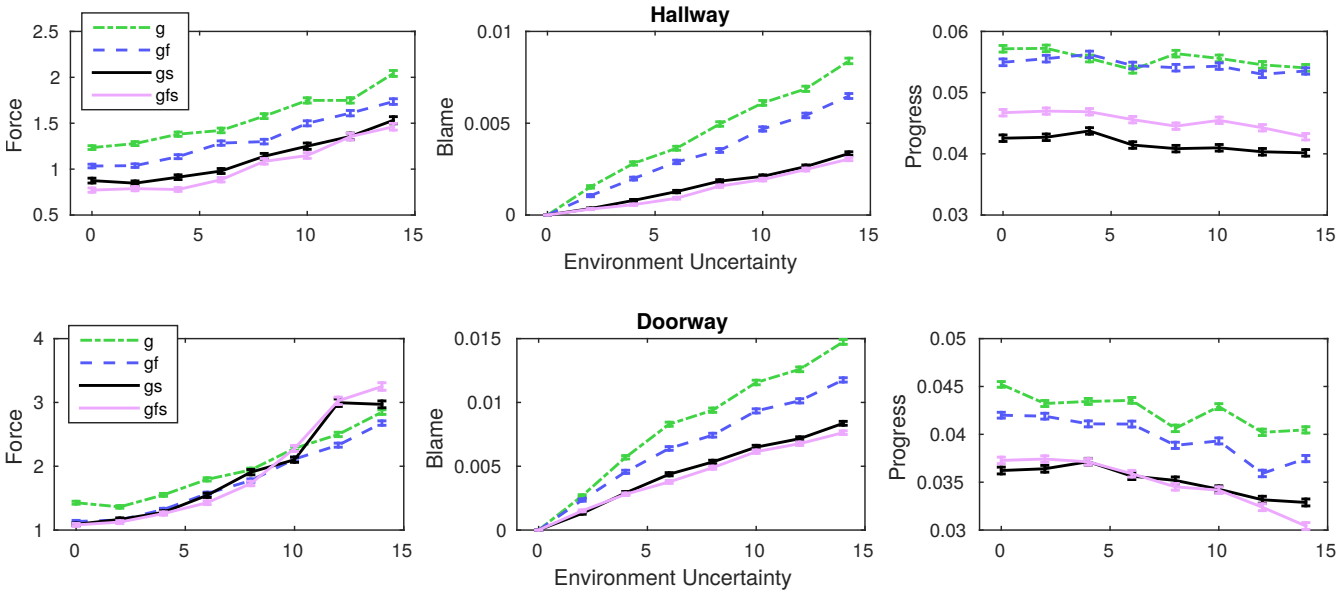


Fig. 5. Simulation results varying uncertainty in the environment (k_z) for a fixed posterior uncertainty (k_e). We show results for 4 combinations of the policies, varying the flexibility of MPDM: *Go-Solo* (g), *Go-Solo* and *Follow* (gf), *Go-Solo* and *Stop* (gs) and the full policy set (gfs). The data is averaged in groups of 10. We show the mean and standard error. *Left:* Increasing the noise in the environment makes the robot more susceptible to disturbing other agents and vice-versa. We can observe that the *Force* when combining all the policies (gfs) is much lower than when using a single policy (g) in the hallway domain. We observe that the robot stops more often in the doorway domain with increasing noise from the declining *Progress* (*Bottom-Right*) for (gs) and (gfs). The high *Force* that arises when the doorway domain is made noisy (*Bottom-Left*) is because the robot can accumulate *Force* in crowded situations even when *Stopped*. This motivates the introduction of *Blame* to study the behavior of the system. *Center:* A lower *Blame* indicates better behavior as the robot is less often the cause of inconvenience. The robustness of MPDM can be observed in milder slope across both domains. *Right:* Higher *Progress* is better. The *Go-Solo* performs better, however at the price of being much worse in *Force* and *Blame*. With more flexibility, (gfs) is able to achieve greater *Progress* and lower *Force* as compared to (gf).

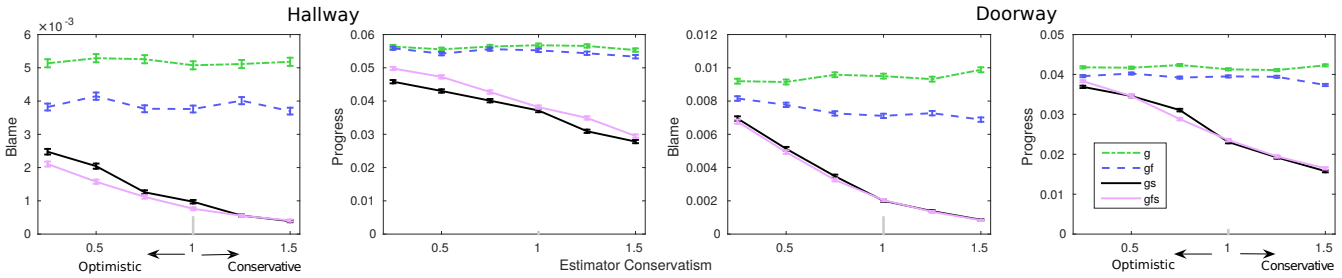


Fig. 6. Navigation performance varying the degree of conservatism ($\frac{k_e}{k_z} = \{0.25, 0.5 \ldots 1.5\}$) of the estimator averaged over $k_z = \{2, 4, \ldots, 14\}$. Combinations of the policies as presented in Fig. 5. The data is averaged in groups of 10. We show here the mean and standard error. The robot errs on the side of caution and *Stops* more often (manifested by a decline in *Progress*) for (gfs) as the robot overestimates the uncertainty in the environment. Additionally, the *Follow* becomes less attractive due to high uncertainty associated with the leader's state. Without the option of *Stopping*, (g) and (gf) maintain high *Progress* and *Blame* which is undesirable since the robot's behavior is indifferent to estimator conservatism and just react to sensory data. With the *Stop* policy (gs,gfs), the robot can adapt to a conservatism of the estimator and can behave cautiously when required.

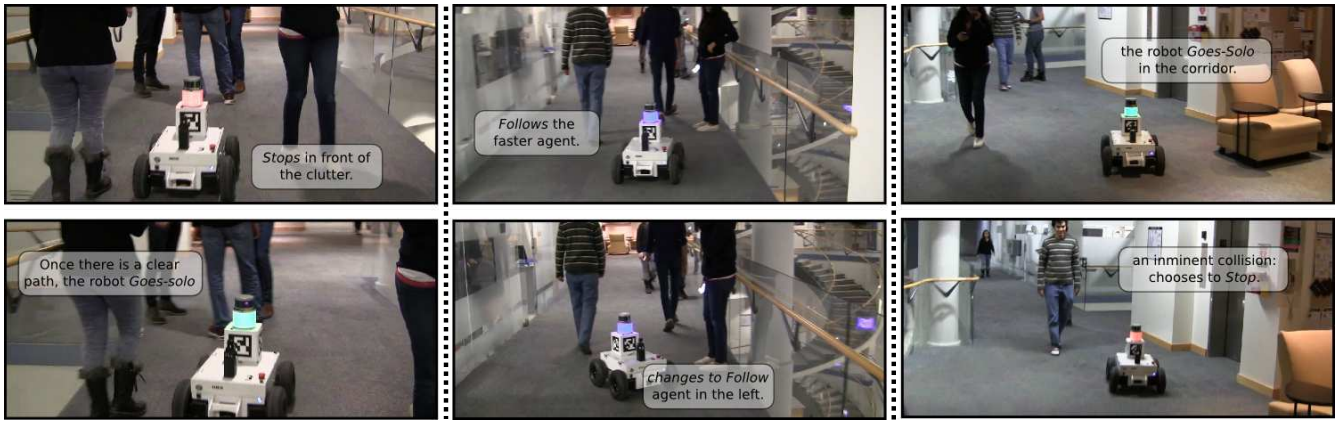
For each setting, we run 100 epochs to collect 30k samples for the metrics.

1) Varying environment uncertainty for a fixed level of estimator optimism: We have studied the impact of different levels of environment uncertainty (k_z) at regular intervals of $\mathrm{diag}(4cm, 4cm, 4cm/s, 6°)$. The estimation uncertainty k_e is fixed at $\mathrm{diag}(10cm, 10cm, 10cm/s, 15°)$.

Fig. 5 shows the performance of the robot for the hallway and the doorway domain respectively. We observe that the *Blame* increases at a lowest rate for MPDM with the complete policy set. If the option of stopping is removed, we notice that the addition of the follow policy allows the robot to maintain comparable *Progress* while reducing the force and *Blame* associated. Given the option of stopping, the robot still benefits from the option of following as it can make more *Progress* while keeping *Blame* and *Force* lower.

We observe that MPDM allows the robot to maintain *Progress* towards the goal while exerting less *Force* and incurring less *Blame*. We also observe that the robot is more robust to noise in terms of *Blame* incurred (lesser rate of increase).

2) Varying the optimism of the estimator: We have studied the impact of different levels of optimism for the estimation error by varying the ratio $\frac{k_e}{k_z}$ from 0.25 to 1.5 in steps of 0.25 for the settings of k_z mentioned above. The ratio indicates over-estimation (> 1) or under-estimation (< 1). For each



(a) (b) (c)

Fig. 7. Real situations (a,b and c) illustrating the nature of the MPDM. On the top row is depicted some situations while testing the robot navigation in a real environment. On the bottom row are shown the same configurations, but delayed by a few seconds. The lights on the robot indicate the policy being executed, being green for *Go-Solo*, blue *Follow* and red *Stop*. By dynamically switching between policies, the robot can deal with a variety of situations.

ratio, we average over the values of k_z.

Figs. 6 shows performance trends for both the domains. In the doorway domain, we notice a lot of queuing behavior near the doorway. For high values of k_z, the robot is very cautious and is often *Stopped* (reason behind the declining *Progress*). Agents come very close to each other even if the robot is stationary. Thus the performance of the system is better captured by *Blame* rather than *Force* as explained in Fig. 5.

We notice that the *Progress* as well as the *Blame* decline as the robot over-estimates the noise and *Stops* more often indicating that we err on the side of caution. On the other hand, a ratio lesser than one implies over-optimism and can cause rash behavior marked by greater *Progress* and *Blame* increases. Even in these situations, the flexibility of multiple policies enables navigation with lower *Blame*.

VII. Real-World Experiments

Our real-world experiments have been carried out in the hallway that the simulated hallway domain was modeled on (Sec. VI). We implemented our system on the MAGIC robot [17], a differential drive platform equipped with a Velodyne 16 laser scanner used for tracking and localization. An LED grid mounted on the head of the robot has been used to visually indicate the policy chosen at any time.

During two days of testing, a group of 8 volunteers was asked to patrol the hallway, given random initial and goal positions, similar to the experiments proposed in Sec. VI. The robot alternated between using MPDM and using the *Go-Solo* policy exclusively every five minutes. The performance metrics were recorded every second, constituting a total of 4.8k measurements.

In Fig. 7 are depicted some of the challenging situations that our approach has solved successfully. On the *Right* and *Left* scenes, the robot chooses to *Stop* avoiding the "freezing robot behavior" which would result in high values of *Blame* and *Force*. As soon as the dynamic obstacles are no longer an hindrance, the robot changes the policy to execute and *Goes-Solo*. In Fig. 7-*Center* we show an example of the robot executing the *Follow* policy, switching between leaders in order to avoid inconveniencing the person standing by the wall. The video provided[1] clearly shows the limitations of the *Go-Solo* and how MPDM solves these limitations.

Fig. 8 shows the results of MPDM compared to a constant navigation policy - *Go-Solo*. As discussed before in Sec. VI, we show that our observations based on simulations hold in real environments. Specifically, MPDM performs much better, roughly 50%, in terms of *Force* and *Blame* while sacrificing roughly 30% in terms of *Progress*. This results in the more desirable behavior for navigation in social environments that is qualitatively evident in the video provided.

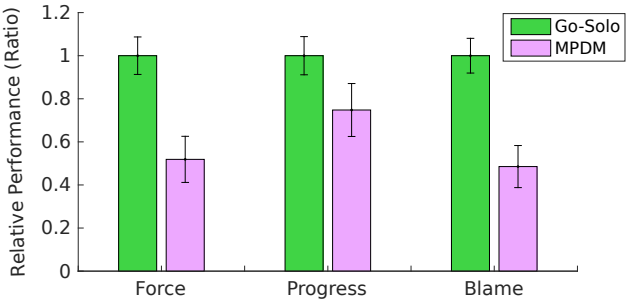


Fig. 8. The mean and standard error for the performance metrics over 10 second intervals (groups of 10 samples) using data from 40 minutes of real world experiments. All measures are normalized based on the corresponding mean value for the *Go-Solo* policy. This figure demonstrates that our results obtained in simulations (Sec. VI) hold on real environments. MPDM shows much better *Force* and *Blame* costs than only *Go-Solo* at the price of slightly reducing its *Progress*.

VIII. Conclusions

In this paper, we have extended Multi-Policy Decision Making (MPDM) , applying it to the robot motion planning

[1] https://www.youtube.com/playlist?list=PLbPJN-se3-QiwIITl5cNsUV4-SRIyl9OM

in real social environments. We show that planning over the policies *Go-Solo*, *Follow-other*, and *Stop*, allows us to adapt to a variety of situations arising in this dynamic domain.

By switching between our proposed set of policies, we have shown that we can improve the performance of the robot as measured by a utility function that rewards task completion (*Progress*) and penalizes inconvenience to other agents (*Force* and *Blame*).

We have shown that reasoning over multiple complementary policies instead of using a single navigation algorithm results in flexible behavior that can deal with a wide variety of situations in addition to being robust to sensor noise and estimator conservatism.

IX. Acknowledgements

The authors are grateful to Robert Goeddel, Carl Kershaw, John Mamish and Justin Tesmer for their help in performing the experiments on the MAGIC robot platform.

References

[1] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proc. IEEE Int. Conf. Robot. and Automation, Seattle, WA, USA*, 2015.

[2] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic motion planning among moving obstacles following typical motion patterns," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4027–4033.

[3] A. Foka and P. Trahanias, "Probabilistic Autonomous Robot Navigation in Dynamic Environments with Human Motion Prediction," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 79–94, 2010.

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

[5] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4293–4298.

[6] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *European Conference on Mobile Robotics*, 2013, pp. 331–336.

[7] ——, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1688–1694.

[8] G. Ferrer and A. Sanfeliu, "Multi-objective cost-to-go functions on robot navigation in dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3824–3829.

[9] W. H. Huang, B. R. Fajen, J. R. Fink, and W. H. Warren, "Visual navigation and obstacle avoidance using a steering potential function," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 288–299, 2006.

[10] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3931–3936.

[11] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[12] M. Luber, L. Spinello, J. Silva, and K. O. Arras, "Socially-aware robot navigation: A learning approach," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 902–907.

[13] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, 2016.

[14] P. Stein, A. Spalanzani, V. Santos, and C. Laugier, "Leader following: A study on classification and selection," *Robotics and Autonomous Systems*, vol. 75, Part A, pp. 79 – 95, 2016.

[15] M. Kuderer and W. Burgard, "An approach to socially compliant leader following for mobile robots," in *International Conference on Social Robotics*. Springer, 2014, pp. 239–248.

[16] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[17] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman and B. Marinier, "Progress toward multi-robot reconnaissance and the magic 2010 competition," *Journal of Field Robotics*, vol. 29, no. 5, pp. 762–792, 2012.

[18] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics Research, Springer Tracts in Advanced Robotics*, vol. 70, pp. 3–19, 2011.

[19] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2009, pp. 177–187.

[20] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

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

[22] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, pp. 1398–1404.

[23] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 1999, pp. 341–346.

[24] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4057–4062.