Multi-Policy Decision Making for Reliable Navigation in Dynamic Uncertain Environments

by

Dhanvin Mehta

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science And Engineering)
in the University of Michigan
2018

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Navigating everyday social environments, in the presence of pedestrians and other dynamic obstacles, remains one of the key challenges preventing mobile robots from leaving carefully designed spaces and entering our daily lives. The complex and tightly-coupled interactions between these agents make the environment dynamic and unpredictable, posing a formidable problem for robot motion planning. 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). Ideally, the robot must account for coupled agent-agent interactions while reasoning about possible future outcomes, and then take actions to advance towards its navigational goal without inconveniencing nearby pedestrians.

Multi-Policy Decision Making (MPDM) is a novel framework for autonomous navigation in dynamic, uncertain environments where the robot's trajectory is not explicitly planned, but instead, the robot dynamically switches between a set of candidate closed-loop policies, allowing it to adapt to different situations encountered in such environments. The candidate policies are evaluated based on short-term (five-second) forward simulations of samples drawn from the estimated distribution of the agents' current states. These forward simulations and thereby the cost function, capture agent-agent interactions as well as agent-robot interactions which depend on the ego-policy being evaluated.

In this thesis, we propose MPDM as a new method for navigation amongst pedestrians by dynamically switching from amongst a library of closed-loop policies. Due to real-time constraints, the robot's emergent behavior is directly affected by the quality of policy evaluation. Approximating how good a policy is based on only a few forward roll-outs is difficult, especially with the large space of possible pedestrian configurations and the sensitivity of the forward simulation to the sampled configurations. Traditional methods based on Monte-Carlo sampling often missed likely, high-cost outcomes, resulting in an over-optimistic evaluation of a policy and unreliable emergent behavior. By re-formulating policy evaluation as an optimization problem and enabling the quick discovery of potentially dangerous outcomes, we make MPDM more reliable and risk-aware.

Even with the increased reliability, a major limitation is that MPDM requires the system designer to provide a set of carefully hand-crafted policies as it can evaluate only a few policies reliably in real-time. We radically enhance the expressivity of MPDM by allowing policies to have continuous-valued parameters, while simultaneously satisfying real-time constraints by quickly discovering promising policy parameters through a novel iterative gradient-based algorithm.

Overall, we reformulate the traditional motion planning problem and paint it in a very different light — as a bilevel optimization problem where the robot repeatedly discovers likely high-cost outcomes and adapts its policy parameters to avoid these outcomes. We demonstrate significant performance benefits through extensive experiments in simulation as well as on a physical robot platform operating in a semi-crowded environment.

CHAPTER 1

Introduction

1.1 Motivation

Navigating dynamic social environments remains one of the key challenges preventing mobile robots from leaving carefully designed spaces and entering our daily lives. Human motion is inherently unpredictable (pedestrians can suddenly stop or change direction without signaling). The complex interactions between pedestrians and the robot give rise to a large number of diverse possible future outcomes. Additionally, the robot must deal with sensor noise and tracking errors. In such an environment, the robot should be able to react to sudden and unexpected changes in the environment; its emergent behavior should be quick, yet reliable.

Research on autonomous navigation began way back in 1966, with Shakey, the world's first mobile robot that could reason about its own actions [1]. Despite decades of progress in the field of motion planning and control, the lack of robots in everyday environments today is a testament to the limitations of the state-of-the-art for navigating dynamic social environments. Many of the commercial deployments today (such as the Kiva warehouse-robot system, or the Patrick shipping terminal at Brisbane) have segregated spaces for robots and humans. Moreover, the robots are often guided by a centralized planning system and need not account for uncertainty while navigating [2].

It is relatively easy to design mobile robots that are slow and safe. However, for emerging applications like autonomous vehicles or delivery robots moving at a high-speed, the navigation algorithm must be able to deal with the large space of possible future outcomes while staying reactive and reliable. While an overly-conservative robot moving slowly can avoid collisions by stopping reactively, not only does it hurt productivity, but the emergent behavior also makes it uncomfortable for humans to co-inhabit such spaces [3]. For example, if an autonomous vehicle drives very slowly on the left-hand lane, it can frustrate other drivers, encouraging them to make a dangerous pass from the right. Similarly,

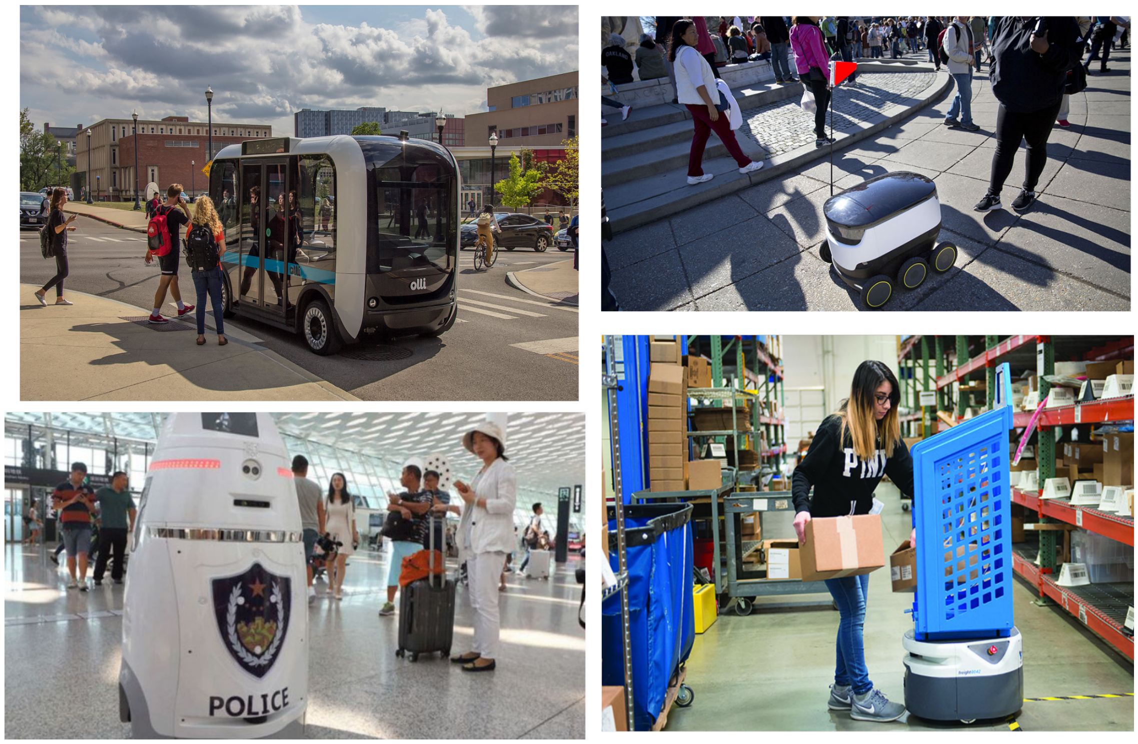


Figure 1.1: Mobile robots in dynamic environments. The ability of a robot to navigate effectively amongst people is critical for services being commercialized today. From *Top-Left* to *Bottom-Right* are pictures of a driverless shuttle Olli developed by Local Motors, a last-mile delivery robot developed by Starship Technologies, a security robot, Anbot, deployed at the Shenzhen airport, and an assistive warehouse robot from Fetch Robotics.

waiting too long at a left turn can cause traffic to back up or can block a crosswalk. In situations like these, it is desirable for the robot to drive more aggressively without compromising safety. This dissertation presents Multi-Policy Decision Making (MPDM), a novel behavioral planning framework to achieve quick and reliable emergent behavior in dynamic, social environments.

The next section introduces the MPDM framework in the context of the popular planning and control architectures for mobile robots that have emerged over decades of research. We begin by revisiting a crucial idea that emerged in the 1980s about how computer programs can produce intelligent behavior. Is it necessary for intelligence to be innate, or is it possible that intelligence is merely an interpretation of a complex system by an observer, thereby existing only in 'the eye of the beholder'?

1.2 Intelligence and Emergent Behavior in Mobile Robots

Representation was believed to be the key to artificial intelligence. The goal was to replicate the human mind in a machine through a central cognitive system (analogous to the brain), which could compute rational actions by reasoning over knowledge obtained through the perception system [4]. Most research focused on assimilating and representing knowledge in an attempt to model the world from a stream of sensory inputs [5]. Unfortunately, modeling the real-world proved to be an extremely ambitious task and progress was slow. Even after three decades, most robots could only operate in static and completely mapped environments [6].

The real-world is full of non-determinism and we cannot predict a priori how it will evolve. Optimizing actions based on imperfect predictions can be counter-productive, especially in a dynamic world where timing is critical [7]. Navigation algorithms that rely heavily on high-fidelity models of the world continue to have limitations even today. Trajectory planning methods [8–10] use models of human behavior to propagate the state of the environment, but they fail to account for the closed-loop coupled interactions between agents. Due to real-time constraints, these methods struggle to incorporate uncertainty into the planning process. POMDPs [9] provide a principled approach to deal with uncertainty, but they quickly become intractable. Even online POMDP solvers such as DESPOT [11], which repeatedly compute optimal short-term policies are limited to small state-action spaces due to the computational burden of reasoning over lots of low-probability outcomes in real-time. Learning-based approaches [12–15] have recently made a lot of progress, but they are limited by the training scenarios considered which might not be a representative set of the diverse situations that may arise in the real world.

At the foundation of behavior-based robotics [7] is the profound realization that perception and action, alone, are sufficient to create the impression of intelligence in a complex system. Through a series of thought experiments, Valentino Braitenberg [16] showed that even extremely simple robots with no cognitive reasoning, could produce seemingly intelligent behavior. Moreover, by progressively adding simpler components, it was possible to build robots that appeared to be aggressive or cowardly, even though these behavioral traits are merely interpretations of an observer trying to analyze the robot's behavior. Simon's Law [17] notes that complex behavior can simply be a reflection of complexity in the environment. For example, on a complex terrain, simple reactive rules can mimic an ant's seemingly complex behavior.

With the idea that intelligent behavior can be synthesized *without* the need for explicit knowledge representation and reasoning, Brooks introduced the subsumption architecture for robotics [18]. The subsumption architecture decomposes the robot's task into layers of simple domain-specific behaviors (such as avoiding obstacles, wandering, or following a wall), where each layer achieves a certain level of competance towards the overall task. Each behavior is self-sufficient and reacts directly to the immediate stimulus *without* the need for a centralized world representation. Different behaviors operate concurrently and their individual responses were coordinated through a hierarchical arbitration scheme.

The 'Creatures' that Brooks built using the subsumption architecture were the most reactive, robust and anthropomorphic mobile robots at that time [6, 19].

1.2.1 Behavior-Based (Reactive) Robotics

Subsumption is just one of several methods have been proposed for resolving the independent responses of behaviors into a single action. The outputs of behaviors can be fused together [20, 21] to produce a combined response, especially when the behaviors are represented as potential-fields [22, 23]. Potential-field based methods became very popular [24, 25] and are still used for navigating dynamic environments even today [26, 27]. However, a reactive potential-field based controller can get stuck in local minima [28] and in dynamic environments, can lead the robot into undesirable configurations due to their short-sightedness [29].

Alternatively, competitive arbitration schemes can determine the 'winning' behavior and thereby, the robot's action in any scenario. Instead of deciding the arbitration scheme a priori, the dominant behavior can be determined at run-time through goal-driven action selection methods [30] or through a contextual voting-scheme [31, 32].

MPDM, like other behavior-based systems, switches between different policies to navigate a social environment. For example, to navigate a crowded corridor, the robot uses the complementary policies illustrated in Fig. 1.2. The robot may *Go-Solo*, moving as fast as possible towards the goal while avoiding collision with pedestrians, or it may *Follow* another person through the crowd, sacrificing speed in order to obtain a clear path, or it may *Stop* in case of mounting uncertainty. However, unlike traditional behavior-based systems, MPDM reasons over a centralized probabilistic representation of the world through multiple forward-simulations in order to choose the optimal policy for a given scenario (Sec. 1.3). Hence, MPDM demonstrates intelligence as a result of innate reasoning as well as emergent behavior from dynamically switching between complementary policies.

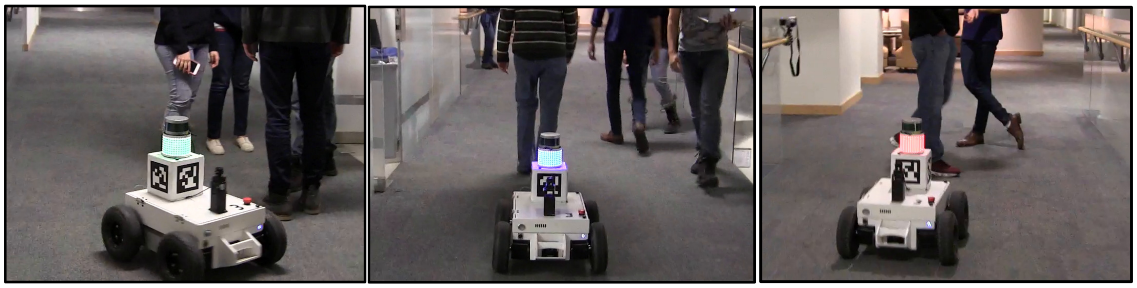


Figure 1.2: MPDM uses a complementary set of closed-loop behavioral policies to navigate a crowded corridor. In MPDM, rather than optimizing trajectories, a planning process chooses from a set of complementary closed-loop policies. The color on the robot's LED ring indicates the elected policy. The robot may choose the (a) *Go-Solo* policy (green), treating pedestrians as obstacles and overtaking them to reach its goal, or it may choose to (b) *Follow* another pedestrian (blue) through the crowd, or it may *Stop* until it has a clear path (red). By switching between different behavioral policies, the robot can adapt to different situations it encounters while navigating a corridor.

1.2.2 Hybrid Planners

As Hanks and Firby [33] noted, two reasonable approaches for motion planning are 1) planning carefully as far ahead of time as possible using rich models of the world and 2) acting only at the last moment to reduce the effect of uncertainty. The two approaches have complementary properties; the former is far-sighted but slower, and relies heavily on model accuracy while the latter is responsive and more robust to uncertain, dynamic environments, but is also myopic. Hybrid architectures attempt to get the best of both worlds by simultaneously executing low-level reactive control and higher-level decision making [34].

Typical hybrid solutions tend to decouple the two systems as much as possible. For example, the low-level reactive process can responsible for obstacle avoidance and immediate safety, while the higher level can take care of global path-planning. As long as the components are decoupled and not in conflict, the hybrid system is easy to design. However, a tighter coupling between the fundamentally disparate parts of the system is challenging and often requires an intermediate coordinating component [35]. Despite a lot of research on the design of such systems [36–39], the way in which the components should be partitioned is still not well understood [7].

The MPDM framework is a novel hybrid-system, where the higher-level process directly models and reasons over possible outcomes from the (lower-level) policies. The elected policy is a reactive controller, responsible for collision-avoidance. This dissertation demonstrates the benefits of our tightly-coupled system for autonomous navigation in dynamic, uncertain environments.

1.2.3 MPDM as a Behavioral Planning Framework

In Multi-Policy Decision Making (MPDM), the robot's trajectory is not explicitly planned, but instead, the robot dynamically switches between a set of closed-loop policies (behaviors), allowing it to adapt to different situations encountered in such environments. Further, each candidate ego-policy is evaluated based on multiple forward-simulations of samples drawn from the robot's probabilistic estimate over not only the current state but also future intentions of nearby agents.

When first introduced, MPDM demonstrated interesting emergent behavior such as merging and overtaking in the context of autonomous driving on highways [40, 41]. However, the lanes and the 'rules of the road' in this domain impose a strong bias on future outcomes and limit the plausible agent-agent interactions, thereby simplifying the problem. Earlier MPDM-systems could only deal with a small discrete set of policies for the robot and other agents.

In this dissertation, we extend MPDM to more dynamic and unstructured social environments, where humans can instantaneously stop or change direction without signaling. The robot must deal with a much larger and more diverse range of possible future outcomes. To address these challenges, we fundamentally overhaul MPDM's core policy election process. We radically improve MPDM's reliability and expressivity, not only enabling the robot to reason over continuous policy spaces for other agents (pedestrians, in our case) but also allowing the robot to choose from an infinite set of ego-policies.

In the following sections, we formulate MPDM and summarize the core ideas and contributions of this thesis that have made MPDM an effective and practical framework for autonomous navigation in dynamic, uncertain environments to navigate in dynamic, unstructured environments.

1.3 MPDM for Navigating Dynamic, Social Environments

MPDM evaluates a "library" of policies (reactive controllers) using an on-line forward roll-out process and the "best" policy is executed until the next planning cycle. Every 300ms, a deliberative policy-election process evaluates each candidate ego-policy based on multiple forward-simulations of samples drawn from the robot's probabilistic estimate over the current state and policies (intentions) of nearby agents. These forward simulations model the coupled interactions between agent behaviors. The chosen policy, however, reacts to each new measurement (every 20ms) which makes the robot highly reactive and agile. By quickly switching between policies, the robot is able to adapt to sudden and unexpected changes in the environment. A robot navigating using MPDM is able to anticipate how scenarios are likely to unfold and modulate its behavior *now* so as to avoid possible hazards in the *future*.

Our model of the environment consists of static obstacles (e.g. walls or doors) and a set of freely moving pedestrians. At any time, agents (the pedestrians as well as the robot) are assumed to be acting according to some policy, which we model as a reactive controller. The system designer is free to choose the set of policies used by the robot to infer pedestrian behavior, as well as the candidate policies for the ego-robot to execute.

For each observed agent i, the robot maintains a probabilistic estimate of its state - i.e. its position, velocity, and inferred policy. The collective state $\boldsymbol{x}_t \in \mathrm{X}$ consists of the state of the robot and all observed agents at time t. Throughout the paper, we will refer to $\boldsymbol{x}_0$ as the collective state of all agents and the robot's state at the beginning of the planning cycle. The robot's probabilistic estimate $P(\boldsymbol{x}_0)$ is based on past observations of the pedestrians' positions. Several methods can be used for obtaining this posterior; in this thesis, we use a Kalman Filter to infer position and velocity and a Naive Bayes Classifier to infer an agent's policy parameters.

During each planning cycle, the robot evaluates candidate ego-policies by forward-simulating several initial configurations sampled based on $P(\boldsymbol{x}_0)$. The agent policies in these forward-simulations capture agent-agent interactions as they try to maintain their desired heading and speed, while simultaneously avoiding collisions. Formally, fixing the candidate ego-policy π_r being evaluated, an initial sampled configuration $\boldsymbol{x}_0$ is forward simulated H time-steps (through $t = 1, \ldots, H$), by recursively applying the transition operator $T : \mathrm{X} \rightarrow \mathrm{X}$ for the time-horizon to yield a trajectory

$$\begin{aligned}\boldsymbol{X}(\boldsymbol{x}_0, \pi_r) &= \{\boldsymbol{x}_0, T(\boldsymbol{x}_0), T^2(\boldsymbol{x}_0), \ldots, T^H(\boldsymbol{x}_0)\} \\ &= \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_H\},\end{aligned}$$

where $\boldsymbol{x}_t \in \mathrm{X}$ is the collective state comprising of the state of the robot plus all the agents at time t of the forward simulation. The operator $T()$ captures the policy that each agent is executing while at the same time considering the interactions with all other agents.

The cost function $C\big(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r)\big)$ assigns a scalar value to the outcome of a forward simulation. We choose a cost function that penalizes the inconvenience the robot causes to other agents in the environment (*Blame*) along the predicted trajectory and rewards the robot's progress towards its goal (*Progress*).

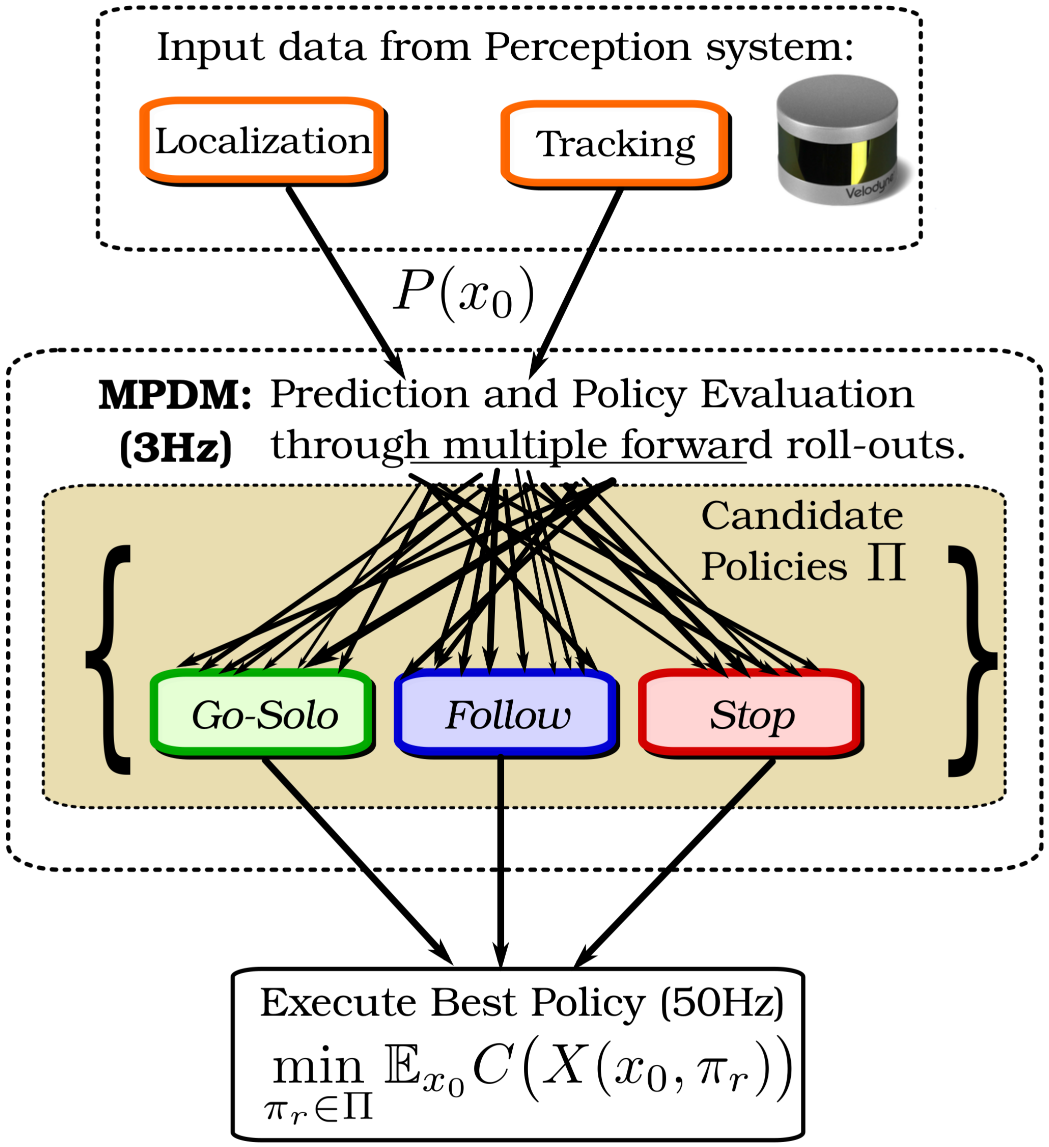

Input data from Perception system:
Localization
Tracking
Velodyne
$P(x_0)$
MPDM: Prediction and Policy Evaluation
(3Hz) through multiple forward roll-outs.
Candidate Policies Π
Go-Solo
Follow
Stop
Execute Best Policy (50Hz)
$\min_{\pi_r \in \Pi} \mathbb{E}_{x_0} C\big(X(x_0, \pi_r)\big)$

Figure 1.3: The Multi-Policy Decision Making framework. The perception system uses of sensor measurements for localization as well as to maintain a probabilistic estimate of the current positions and velocities for tracking nearby pedestrians. Each ego-policy is evaluated based on numerous forward simulations of samples drawn from this distribution. The policy with the best expected utility is chosen and executed.

Every planning cycle (3Hz), the ego-robot is tasked with determining the 'best' policy. Traditionally [40], the ego-policy with the best *expected utility* is chosen for execution.

$$\pi^* = \underset{\pi \in \Pi}{\arg\min}\, E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r)\big)\} \tag{1.1}$$

Chapter 3 introduces MPDM as a a new method for navigation amongst pedestrians using complementary 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.

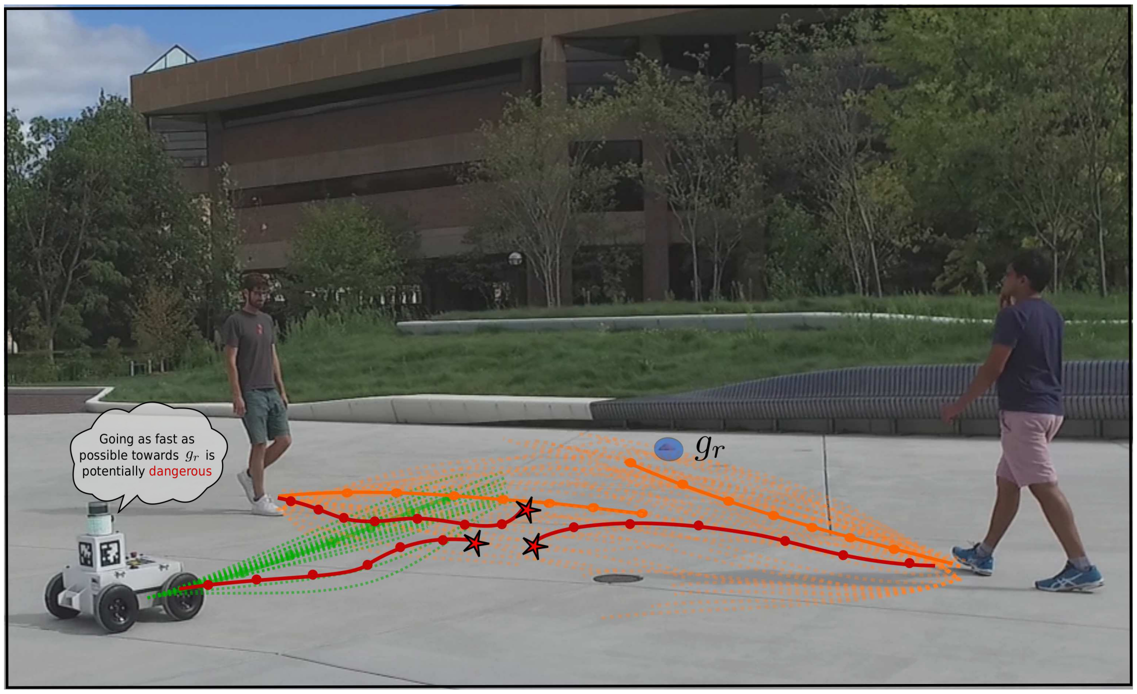
1.4 Discovering Influential Outcomes

The robot's uncertainty about the inferred state of the dynamic agents (pedestrians) and the complex multi-agent interactions make the forward-simulated trajectories (Fig. 1.4) sensitive to the initial configurations sampled, resulting in a wide range of possible future outcomes. In order to stay reactive to sudden and unexpected changes in the environment (e.g. a person suddenly jumping in front of the robot), MPDM must re-plan frequently. Therefore, the robot must evaluate a policy reliably in as few forward simulations as possible.

Earlier approaches [40] approximated each policy's expected utility (Eqn. 1.1) by forward-simulating a few randomly sampled (Monte-Carlo) initial configurations. However, since the policies that the agents are executing in the forward simulation try to avoid collision, near misses are mundane and collisions, although profoundly serious, are rare. Sampling randomly is therefore likely to miss high-cost events, even if they are individually reasonably probable (high probability density) because of the scarcity of such configurations in the state space (low total probability mass of high-cost outcomes). Monte-Carlo sampling is agnostic to the cost function and thus, fails to consistently capture the important (high-cost) configurations that should ideally impact the utility estimate.

Discovering these configurations through random sampling may require drawing many samples, which becomes a performance bottleneck. Without enough samples to find influential outcomes, the quality of planning suffers. Addressing this issue is crucial for reliable systems in applications such as autonomous cars, navigation in social environments, etc.

In chapter 4, the main idea is that rather than relying on random sampling, casting policy evaluation as an optimization problem enables quick discovery of influential outcomes


Going as fast as possible towards g_r is potentially dangerous
g_r
Figure 1.4: An illustration motivating risk-aware planning. The trajectories arising from the most likely initial configuration for the agents (orange) and the robot (green), like most outcomes of possible initial configurations (dashed lines) are benign. Near misses are mundane as the agents in the forward simulation tend to avoid collision. As a result, there may be a few dangerous initial configurations that may be individually likely and yield high-cost outcomes (red). While evaluating policies, likely collisions (red stars) should be discovered as quickly as possible to allow larger candidate policy sets for MPDM.

by biasing sampling towards increasingly likely and high-cost outcomes. We define *influential* outcomes as high-cost outcomes that are likely based on the robot's belief $P(\boldsymbol{x}_0)$. We reformulate the core optimization, evaluating each candidate policy based on the *most influential outcome*.

$$\pi^* = \arg\min_{\pi \in \Pi} \max_{\boldsymbol{x}_0} \{P(\boldsymbol{x}_0) C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)\}. \tag{1.2}$$

Finding good solutions to the optimization problem Eqn. 1.2 in real-time is still challenging due to the large number of possible outcomes as well as the sensitivity of the forward simulation to the initial configuration. In chapter 4, we show that accurate gradients can be computed – even through a complex forward simulation – using approaches similar to those in deep networks. Deep neural networks model complex functions by composing (chaining) relatively simple functions (convolutions or ReLU modules). Similarly, a forward simulation captures the complex dynamics of the system using simple one-step transition functions T (Fig. 1.5). Since our cost function is a linear combination of costs computed along the trajectory, we can conceptualize the forward simulation as a deep net-

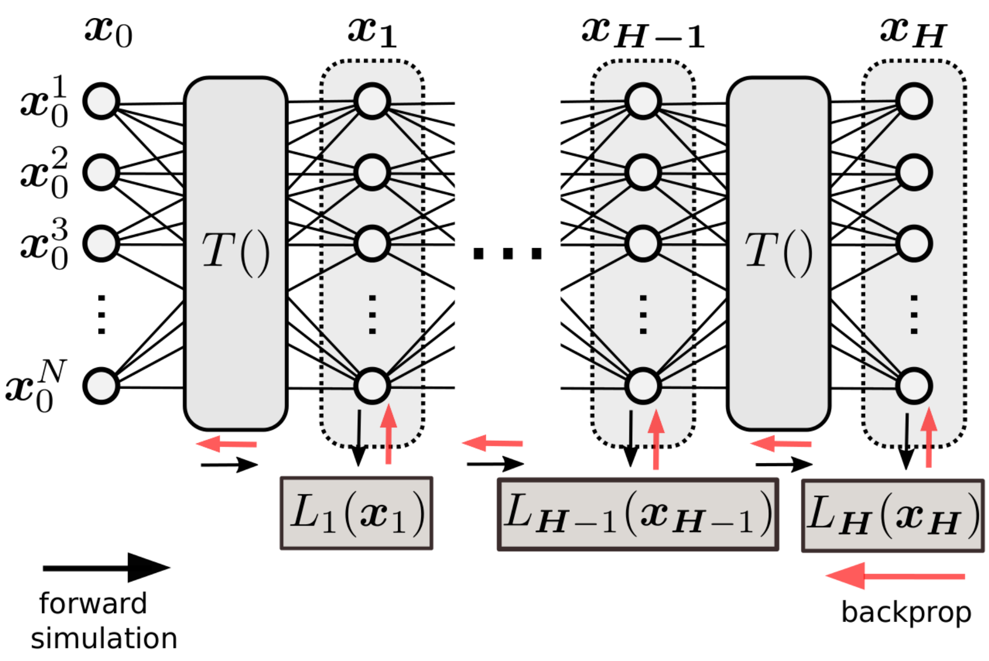


Figure 1.5: A deep network representation for our cost function. The initial configuration $\boldsymbol{x}_0$ propagates through several layers, each representing the transition function T. The output of layer t determines a cost $L_t(\boldsymbol{x}_t)$. Our cost function $C(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0))$ accumulates costs calculated at each time-step along the forward simulated trajectory.

work that outputs a trajectory cost $C(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0))$ based on the input initial configuration. We demonstrate significant performance benefits of risk-aware MPDM in simulation as well as on a real robot platform navigating a highly dynamic environment.

1.5 Continuously Parameterized Policies

Even after improving the efficiency of policy evaluation, only a handful of policies can be evaluated reliably in real-time. It is desirable to add more policies to the system to increase the flexibility of the system, however, this increases computational cost. The ultimate goal of MPDM is to choose the policy with the most benign dangerous outcome and restricting MPDM to a small set (perhaps 5-10) of discrete policies is a significant performance bottleneck. Further, the need for carefully hand-crafted ego-policies also limits the application of MPDM.

In chapter 5, we radically enhance the expressivity of MPDM by allowing policies to have continuous-valued parameters, while simultaneously satisfying real-time constraints by quickly discovering promising policy parameters through a novel iterative gradient-based algorithm, as illustrated in Fig. 1.6. This removes the need for carefully hard-crafted

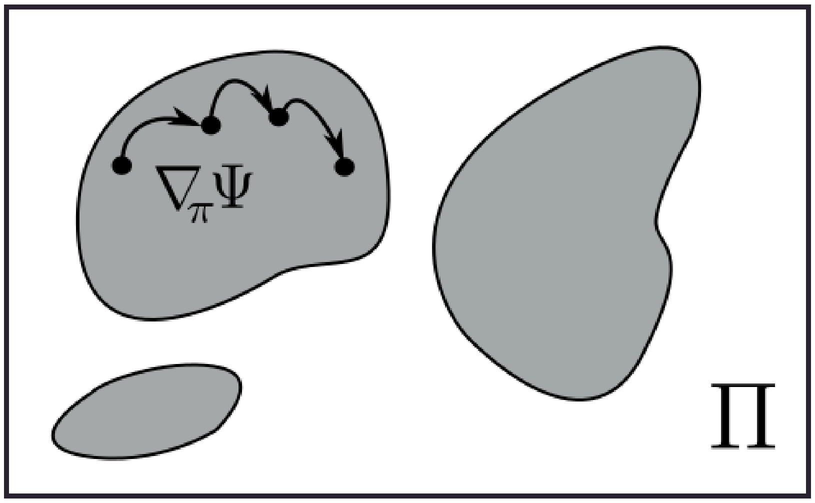


Figure 1.6: Continually-parameterized MPDM (C-MPDM) can represent much larger volumes within the policy space. By quickly generating promising context-derived candidate policies using "risk-aware policy-gradients" $\nabla_\pi \Psi$, C-MPDM increases expressivity of the actions available to the robot without increasing computational complexity.

candidate policy sets for the ego-robot, making MPDM a more flexible and powerful framework for real-time, risk-aware behavioral planning.

Bilevel optimization is a well-studied class of mathematical programs encountered in various fields ranging from management [42], to optimal control [43] where there are two levels of optimization tasks, one nested within the other. In risk-aware MPDM (Eqn. 1.2), the upper-level optimizer (the ego-robot) chooses the policy with the most benign (low-cost) evaluation, while lower-level optimization (risk-aware policy evaluation of an ego-policy) involves finding the most potentially dangerous (likely, high-cost) outcome from all possible pedestrian configurations. In this way, risk-aware MPDM can be viewed as a bilevel optimization, but we now consider an infinite number of ego-policies for the upper-level optimization.

In order to extend MPDM to continuous ego-policy spaces, we need new optimization strategies. Our novel anytime algorithm leverages the layered representation of the forward simulation developed in chapter 4 to find increasingly desirable contextual ego-policy parameters. The intermediate results of our iterative gradient-based procedure can be used to discover influential policy candidates customized to any specific real-time context. Through extensive experiments in simulation and on a real robot platform, we demonstrate the benefits of C-MPDM over two other approaches of evaluating a continuous policy space: a fixed set of hand-crafted policies and random policy sampling.

1.6 Contributions

This thesis radically improves the Multi-Policy Decision Making (MPDM) framework for reliable navigation in dynamic uncertain environments by reformulating the traditional motion planning problem and painting it in a very different light. Our main contributions are as follows -

1. 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) [40] 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.

2. We formulate a risk-aware objective for evaluating policies to bias the sampling of initial configurations towards likely, high-cost (influential) outcomes. This formulation casts the problem of policy evaluation as an optimization problem. We show that our forward simulation can be encoded in a differentiable deep network and propose an efficient procedure using backpropagation to compute an accurate gradient of the objective function without the need for heuristics (Sec. 4.3). Our anytime optimization algorithm finds increasingly influential configurations at every iteration.

3. We exploit the aforementioned layered architecture to solve the notoriously difficult minimax program that underlies risk-aware planning (i.e., minimizing the worst-case cost) via gradient propagation without being restricted to a fixed set of predetermined policies. Thereby, we radically enhance the expressivity of MPDM by allowing policies to have continuous-valued parameters, while simultaneously satisfying real-time constraints by quickly discovering promising policy parameters through the novel iterative gradient-based algorithm.

4. We quantify the improvements made by each of these contributions through extensive experiments in simulation as well as in the real-world on our robot platform. Finally, we compare the performance of C-MPDM with a state-of-the-art trajectory based planner, Model-Predictive Equilibrium-Point Control (MPEPC) [44] through extensive real-world experiments using objective metrics and subjective feedback. We demonstrate that C-MPDM produces emergent behavior that is more reliable than MPEPC for autonomous navigation in dynamic uncertain environments.

CHAPTER 2

Background

The objective of a navigation system is to guide a robot from its current pose (position and orientation) to a target position (goal). In most cases, the target pose is beyond the robots current sensor horizon, and the robot must be aware of the goal location relative to its current pose in some global reference frame. Typically, navigation systems employ a map to support planning outside the robots limited field of view, requiring the robot to be localized within that map. The robot must perceive pedestrians or other dynamic obstacles within the robot's sensor horizon, predict possible future outcomes and pick a suitable action to progress towards its goal, while avoiding inconvenience to nearby pedestrians. Rather than relying only on the current observation, the perception system often tracks pedestrians and maintains an estimate of their current positions and velocities. These estimates may be augmented with domain knowledge such as social norms or salient map locations [45] to bias predicted outcomes.

A typical autonomous navigation system can be broken down into three modules that interact with one another. The *perception module* is responsible for tracking dynamic obstacles and localizing the robot within a mapped environment. It maintains probabilistic state estimates for the robot as well as nearby dynamic agents. Based on these probabilistic estimates, the *planning and prediction module* forecasts possible outcomes and decides the plan of action. Since the robot's actions often affect future outcomes, the prediction and planning sub-systems are tightly coupled. The *lower-level controller module* is responsible for executing the chosen plan (e.g. following the planned trajectory, or moving towards a planned waypoint). The kinodynamic constraints of a robot, as well as the lower-level control, determine how accurately a certain plan can be executed. A high-fidelity controller may not always be available.

The field of perception for mobile robotics has rapidly evolved over the last two decades. Recently, large realistic data-sets such as KITTI [46] have pushed the boundaries of structured prediction and deep learning algorithms for robotic perception. Today, data-driven algorithms have become state-of-the-art for object recognition and tracking [47].

Despite the advances in real-time multi-object tracking [47], predicting future outcomes in social environments remains challenging. Human motion is unpredictable – humans can suddenly change direction or stop without signaling. In addition to the robot's uncertainty about the velocities and future intentions of pedestrians, the complex agent-agent interactions result in a large number of diverse possible future outcomes. In section 2.1, we highlight different approaches to predicting pedestrian motion and agent-agent interactions.

While the accuracy of prediction plays an important role in navigation, a planning algorithm must be able to reason over prediction uncertainty. Motion planning in dynamic uncertain environments still remains a major challenge. Section 2.2 discusses some influential ideas that have helped shape modern motion planning approaches.

2.1 Predicting Pedestrian Motion

Understanding the unwritten rules of pedestrian behavior, and predicting their future motion is a major challenge for autonomous navigation in social environments. Wolpert and Ghahramani showed that intentions precede movement [48].

Some approaches assume a dynamic model of the obstacle and propagate its state using standard filtering techniques such as the extended Kalman filter [49, 50]. Despite providing probabilistic estimates over an obstacle's future states, these methods often perform poorly while dealing with nonlinearities in the assumed dynamics model and the multi-modalities induced by discrete decisions. These methods also fail to capture the complex interactions between the dynamic agents.

Pedestrian trajectories can be clustered with the assumption that people move towards salient points in the environment and the inferred patterns can guide prediction [51–53]. Turek *et al.* [54] use similar ideas to infer functional categories of objects in the environment from tracking objects in a video stream. Various factors like social affinity [55] and human-attributes [56] have been shown to influence the accuracy of motion prediction. Arechavaleta *et al.* [57] propose a geometric criteria governing human motion; humans minimize the variation in path curvature. More recently, personality traits and social norms have been used to predict long-term paths and time-varying behaviors for each pedestrian in real-time [58].

With better hardware, recent progress in deep learning for computer vision, and access to more data, model-based learning algorithms have become state-of-the-art for pedestrian motion. Here, the underlying model implicitly or explicitly encodes latent pedestrian intentions. The assumption that humans move based on some underlying optimal planning algorithm has proven to be effective for generalization. Observed trajectories of humans are used to infer model parameters or cost functions for an underlying decision process [13–15]. Ziebart *et al.* [12] use maximum entropy inverse reinforcement learning (IRL) to learn a cost function for goal-directed pedestrian motion. Kitani *et al.* [59] used a latent variable Markov Decision Process to model interactions and infer traversable paths. Recently, Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) have produced improved models for contextual socially aware trajectory prediction [60–64]. However, it has been shown that many of these models only capture interactions caused by agents that are close-by [65].

In general, the efficacy of learning-based methods 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. Capturing interactions between pedestrians and the robot remains a major challenge for these approaches because such data is much harder to obtain and generalize from.

This thesis explores decision making under prediction uncertainty where there is an important trade-off between prediction accuracy and computational complexity that must be considered. In the field of neuroscience, Helbing and Molnár [23] proposed the Social Force Model, a potential field based approach that describes the interactions between pedestrians in motion. that has proven to be simple, yet effective modeling human-human interactions [66]. High-fidelity predictive models produced by learning-based methods often have a much higher computational cost [67] but their application within MPDM is beyond the scope of this thesis.

2.2 Planning in Dynamic Environments

2.2.1 Classical Motion-Planning Based Approaches

Despite the probabilistic nature of the anticipation problem, several motion-planning methods in the literature assume no uncertainty on the future states of other participants [68–70]. In a simulated environment, van den Berg *et al.* [71] 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.* [72] extended this work using *finite-time velocity obstacles* to provide a locally collision-free solution that was less conservative as compared to van den Berg *et al.* [71].

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. This could be justified in a scenario where vehicles broadcast their intentions over some communication

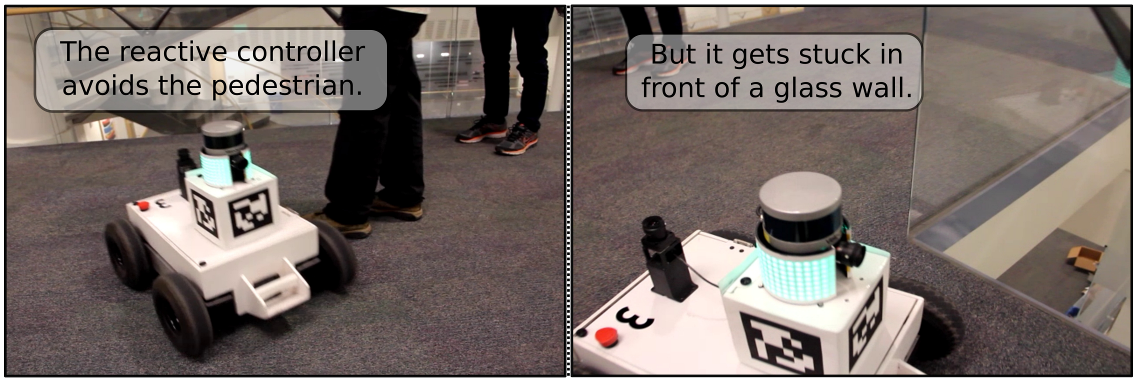

The reactive controller avoids the pedestrian.
But it gets stuck in front of a glass wall.

Figure 2.1: Limitations of a single potential field based policy. Here the robot tried to avoid the oncoming pedestrian (*Left*), but in doing so, gets stuck in front of a glass wall (*Right*).

channel, but it is an unrealistic assumption for social environments.

2.2.2 Reactive Planners

Purely reactive planners pick an action only based on the current positions and velocities of observable pedestrians without predicting possible future interactions between the dynamic agents. Such approaches have the advantage of being very agile, but being myopic, the robot may get stuck in local optima or may take actions that result in undesirable outcomes (Fig. 2.1).

Several approaches attempt to navigate in social environments by traversing a potential field [73] generated by a set of pedestrians [26, 27, 74]. Huang *et al.* [75] used visual information to build a potential field to navigate.

Unfortunately, potential field approaches have some limitations, such as local minima or oscillation under certain configurations [28]. These limitations can be overcome to a certain degree by using a global information plan to avoid local minima [76]. 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. However, rather than using a single potential-field based policy, in MPDM, the robot dynamically switches between a set of policies in order to adapt to different situations it may encounter.

2.2.3 Trajectory Optimization Over Longer Time Horizons

More recent approaches have addressed the decision-making problem for autonomous driving through the lens of trajectory optimization [77–80], but these methods do not model the closed-loop interactions between vehicles, and therefore fail to reason about their potential

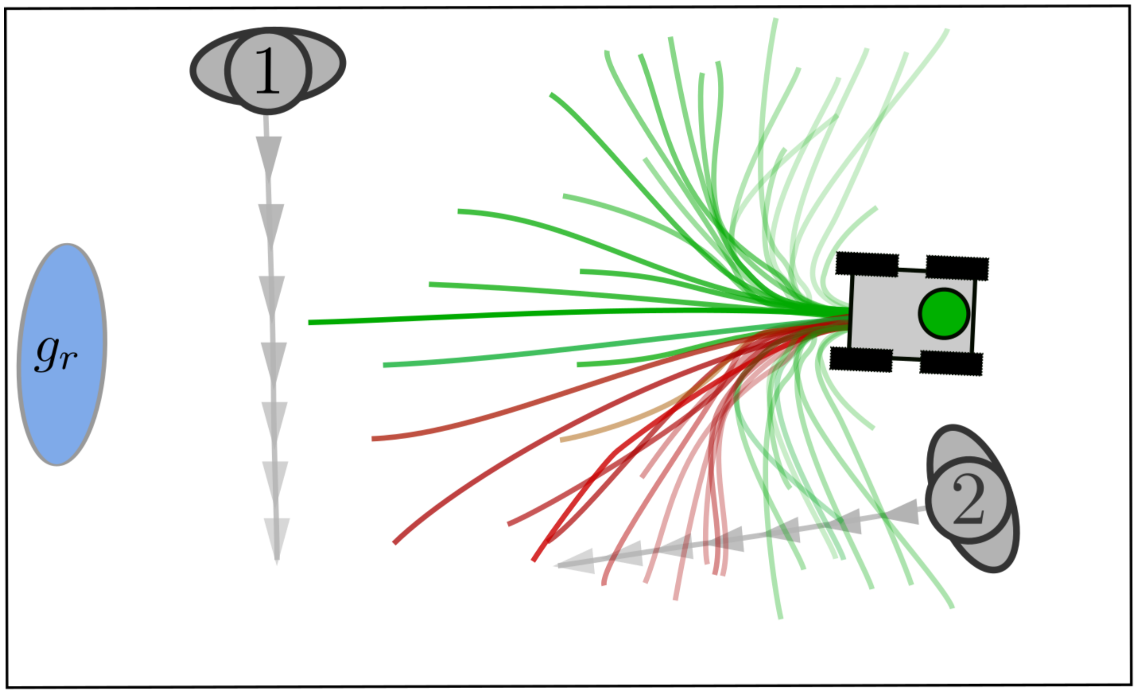


Figure 2.2: Evaluated trajectories from Model-Predictive Equilibrium-Point Control (MPEPC) [44], a state-of-the-art trajectory optimization technique. A large number of robot trajectories are scored in real-time and the trajectory with the least cost is executed. The red trajectories indicate high-cost while the green trajectories signify low-cost. Due to real-time constraints, MPEPC predicts the pedestrian trajectories based on only the most-likely estimated position and velocity of nearby agents. MPEPC does not explicitly account for uncertainty in the estimated velocity or future intentions of the pedestrians. Moreover, such methods require a high-fidelity motion model of the robot such that the robot can execute the planned trajectory.

outcomes. In the context of social navigation, trajectory planning methods [8, 9] use models of human behavior to predict future states of the environment, but may fail to account for the coupled interactions between pedestrians and the robot. Furthermore, MPC-based approaches often rely on a high-fidelity forward simulator which may not be available.

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 [8, 9, 12–15, 26, 27, 74, 75, 81–84] 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. Many of these methods for safe navigation in social environments relied on slow-moving robots, delegating the responsibility of avoiding collision to people.

Model Predictive Control (MPC) is a receding-horizon control algorithm designed for the online-optimization of constrained linear-time invariant systems. While MPC can handle complex constraints, the online optimization may get stuck in local minima in high-dimensional domains [85]. In practice, it is hard to design a smooth cost function that results in good navigation behavior across a wide variety of situations [10]. A major limitation of the MPC framework is that it does not directly handle uncertainty [86]. In dynamic environments, this becomes a major performance bottleneck.

Model Predictive Equilibrium Point Control (MPEPC) is a stochastic MPC approach that address this limitation by incorporating uncertainty into the cost function [10]. MPEPC significantly improved the state-of-the-art in stochastic model predictive control and sample-based optimal path planning, enabling safe, and comfortable automomous navigation in dynamic environments even at high speeds [44]. In MPEPC, several trajectories generated from a high-fidelity robot model are quickly evaluated and the best trajectory is selected. However, it considers only the single most likely outcome predicted based on most-likely esimates of the current position and velocity of nearby agents as well as static obstacles.

Our proposed approach, MPDM evaluates fewer policies, but each policy is evaluted more extensively based on multiple forward simulations that directly capture agent-agent interactions.

2.2.4 Learning-Based Approaches

By observing pedestrians navigate through crowds and using data from expert demonstrations typically obtained through teleoperation, the goal of learning-based methods is to infer navigation policies can emulate human motion.

Gaussian Process (GP) regression has been utilized to learn typical motion patterns for classification and prediction of agent trajectories [87–89], particularly in autonomous driving [90–92].

Learning an underlying cost function from human demonstrations using inverse reinforcement learning (IRL) has proven to be an effective method for tasks such as vehicle parking [93] and driving [94]. Henry *et. al.* [95] extended these methods to partially observable environments and demonstrated that the robot was able to successfully execute human-like behavior in a realistic crowd flow simulator. Thereafter, several promising IRL methods have demonstrated socially compliant navigation on a real robot platform [15, 96, 97]. For example, in the context of autonomous driving, Kuderer *et. al* [98] use inverse reinforcement learning to learn driving styles from trajectory demonstrations in terms of engineered features. They then use trajectory optimization to generate trajectories for their autonomous vehicle that resemble the learned driving styles.

One major concern about feature-based IRL is that features calculated along the paths of pedestrians can vary significantly with different runs even on a similar scenario [96]. This limits the generalizability of these approaches.

Rather than rely on hand-crafted features, end-to-end deep reinforcement learning has received a lot of attention and has been used to learn a navigation policy based on raw sensor inputs [99–101]. Chen *et. al.* [67] define a reward function to learn socially-compliant policy from teleoperation data. More recently, generative adversarial imitation learning [102] has been used for behavior cloning [103].

Nonetheless, these methods require the collection of training data to reflect the many possible motion patterns the system may encounter, which can be time-consuming. Data collection for learning policies in a dynamic social environment is particularly challenging because the robot's actions can significantly affect outcomes. This greatly limits the number of real-world samples that can be collected [104]. Furthermore, in a crowded environment, even small changes in pedestrian intentions and configurations can result in very different outcomes. The collected data might not be a representative set of the diverse situations that may arise in social environments. In this thesis, we allow domain knowledge to be incorporated into the navigation framework by allowing the robot to choose from a "library" of hand-engineered policies.

2.3 MPDM Approximates a POMDP

Early instances of decision-making systems for autonomous vehicles capable of handling urban traffic situations stem from the 2007 DARPA Urban Challenge [105]. In that event, participants tackled decision-making using a variety of solutions ranging from FSM [106] and decision trees [107] to several heuristics [108]. However, these approaches were tailored for specific and simplified situations and were, even according to their authors, "not robust to a varied world" [108].

Partially Observable Markov Decision Processes (*POMDPs*) provide a principled approach to deal with uncertainty, but they quickly become intractable [109, 110]. Various general *POMDP* solvers seek to approximate the solution [111–114]. For example, in the context of autonomous driving, a point-based *MDP* was used for single-lane driving and merging [115]. An MDP formulation employed by [116] for highway driving use *behaviors* that react to other objects, similar to MPDM's *policies*,. Our domain has a high-dimensional belief space with a large number of low-probability outcomes (the curse of dimensionality) which is a major hurdle for POMDP solvers. These methods typically take several hours for problems with much smaller state-action spaces than those encountered in real-world scenarios [117], there has been some recent progress that exploits GPU parallelization [118].

Recently, the idea of assuming finite sets of policies (intentions) to speed up planning has been explored [11, 116, 119–124]. However, these approaches are limited to discrete state-action spaces and use significant resources computing policy sets online, and are therefore limited to short planning horizons. Rather than learn policies, our approach (Multi-policy decision making) exploits domain knowledge to design a set of policies that are readily available at planning time. For example, one approach for navigating in social environments is to look for a pedestrian leader to follow, thus delegating the responsibility of finding a path to the leader, such as the works of [82, 83, 125]. In MPDM, *Follow* becomes one of the policies that the robot can choose to execute as an alternate policy to navigating.

MPDM [40] is a constrained POMDP solver, in which the space of policies that can be evaluated is narrowly constrained by design. In this section, we discuss the series of relaxations that allows MPDM to solve large POMDPs effectively in real-time, while accounting for agent-agent interactions and a large space of possible outcomes.

Let P denote the set of agents near the robot including the robot. At time t, an agent $i \in P$ can take an action $a_t^i \in \mathrm{A}^i$ to transition from state $x_t^i \in \mathrm{X}^i$ to x_{t+1}^i. As a notational convenience, let $x_t \in \mathrm{X}$ include all state variables x_t^i for all agents at time t, and similarly let $a_t \in \mathrm{A}$ be the actions of all agents.

We model the agent dynamics with a conditional probability function capturing the dependence of the dynamics on the states and actions of all the agents in the neighborhood.

$$T(x_t^i, a_t, x_{t+1}) = p(x_{t+1}^i | x_t, a_t). \tag{2.1}$$

Similarly, we model observation uncertainty as

$$Z(x_t, z_t^i) = p(z_t^i | x_t), \tag{2.2}$$

where $z_t^i \in \mathrm{Z}^i$ is the observation made by agent v at time t, and $z_t \in \mathrm{Z}$ is the vector of all sensor observations made by all agents.

The robot's goal is to find an optimal policy π^* that maximizes the expected sum of rewards over a given decision horizon H, where a policy is a mapping $\pi : \mathrm{X} \times \mathrm{Z} \rightarrow \mathrm{A}$ that yields an action from the state and an observation:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=t_0}^{H} R(x_t, \pi(x_t, z_t^r))\right], \tag{2.3}$$

where $R(\cdot)$ is a real-valued reward function $R : \mathrm{X} \times \mathrm{A} \rightarrow \mathbb{R}$.

The evolution of $p(x_t)$ over time is governed by

$$p(x_{t+1}) = \iiint\limits_{\mathrm{X\,Z\,A}} p(x_{t+1}|x_t, a_t)p(a_t|x_t, z_t)p(z_t|x_t)p(x_t)\, da_t\, dz_t\, dx_t. \tag{2.4}$$

Marginalizing over such a large state, observation, and action space is too expensive to find an optimal policy online in a timely manner. A possible approximation to speed up the process, commonly used by general *POMDP* solvers [111, 114] is to solve Eq. 2.3 by drawing samples from $p(x_t)$. However, sampling over the full probability space with random walks yields a large number of low probability samples wasting a lot of computational resources on unlikely outcomes. MPDM samples more strategically from high likelihood scenarios to ensure computational tractability.

The Multi-Policy Approximation

We make the following approximations to sample likely interactions of agents:

1. At any given time, both the robot and other agents (pedestrians) are executing a policy from a set of closed-loop policies. The robot maintains a probabilistic estimate of each observed agent's policy $p(\pi_t^i|x_t, \mathbf{z_{0:t}})$. It is assumed that the agents do not collaborate and for any pair of agents i and j, $p(\pi_t^i|x_t, \mathbf{z_{0:t}})$ and $p(\pi_t^j|x_t, \mathbf{z_{0:t}})$ are uncorrelated.

2. We approximate the motion and observation models through deterministic, closed-loop forward simulation of all agents with assigned policies. The modeled agents react to nearby agents via z_t^i. For each sampled initial configuration (including policies), the forward simulation yields a sequence of future states and observations that incorporates agent-agent interactions. The utility of the forward simulation captures the reward function over the entire decision horizon.

POMDP solvers typically reason over lots of low probability outcomes, which is computationally very intensive. Our approximations allow us to evaluate the consequences of our decisions over a set of high-level behaviors determined by the available policies rather than performing the evaluation for every possible control input of every agent.

The key idea we leverage is that, rather than plan nominal trajectories, we can think of behavior as emerging from choosing closed-loop policies (policies that react to the presence of other agents, in a coupled manner) For instance, in indoor social environments, the robot can plan in terms of going towards the end of a hallway at a certain speed, or following other agents or stopping. Typical behaviors that conform to these rules can greatly limit the action space to be considered and provides a natural way to capture closed loop interactions. Even though we assume a deterministic transition model, MPDM incorporates uncertainty over the observed agent states (including their intents/policies). Given samples from $p(\pi_t^i | x_t, \mathbf{z_{0:t}})$ that assign a policy to each agent, and a policy π_r for the robot, we simulate forward both the robot and the other agents under their assigned policies to obtain sequences of predicted states and observations. We evaluate the expected utility using these sample roll-outs over the entire decision horizon in a computationally feasible manner. Recently, approximate POMDP methods based on scenario sampling and forward simulation have been applied to navigation [126] and mapping [127].

MPDM decouples closed-loop low-level control (implemented by the individual policies) from high-level planning (policy election) by approximating policy outcomes as deterministic functions of an uncertain state, and reduces the POMDP into the following decision making process:

$$\pi^* = \arg\min_{\pi_r} E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)\}, \tag{2.5}$$

where the cost $C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)$ is associated with the current state $\boldsymbol{x}_0$ (which includes the latent intents of other pedestrians) upon choosing a policy π_r.

Cunningham *et al.* [40] and Galceran *et al.* [128, 129] originally formulated MPDM for making decisions based on coupled interactions between cars in a tractable manner. They used a discrete set of lane-changing policies, each capturing a high-level behavior and intention for autonomous driving in highway traffic situations such as driving along a lane or turning at an intersection.

We explore more unstructured social environments where the number of complexity of policies is far greater. In our setting, the dynamic agents (humans) can instantaneously stop or change direction without signaling. Maneuvering in social environments is very 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.

Optimization Techniques

While the aforementioned relaxations allow MPDM to switch between a handful of ego-policies, for effective navigation in dynamic environments, we must address two fundamental issues -

1. MPDM has to evaluate ego-policies under a lot of uncertainty (the dimensionality of the sample space scales linearly with the number of pedestrians considered) in real-time. Since forward-simulation is computationally expensive, an ego-policy must be evaluated using forward roll-outs from a few samples. How can we efficiently sample from the space of all possible outcomes so as to reliably evaluate ego-polices?

2. While it is desirable to add more policies to the system to increase the flexibility of the MPDM system, this increases computational cost. Can we allow for continuous ego-policies, while still maintaining tractability?

Chapters 4 and 5 of this thesis resolve these issues by casting policy evaluation as an optimization problem, and applying novel representations and optimization techniques for risk-aware navigation.

In MPDM, each policy is evaluated through a forward roll-out process. We can enabling the accurate computation of gradients even through a complex forward simulation – using approaches similar to those in deep networks (a forward simulation captures the complex dynamics of the system using simple one-step transition functions). Thus, a new set of tools– well-developed in other disciplines– is brought to bear on behavioral planning. In this section, we discuss some gradient-based methods and other applications of bilevel optimization that have motivated our approach.

A core tool in our proposed work is gradient-based function optimization. These gradients are computed numerically using back propagation methods that closely resemble those used in the training of neural networks. Backpropagation has been the de-facto method for parameter optimization in neural networks since the 80's. Seminal works [130, 131] showed encouraging results using neural networks to learn control actions from perception. With increased computational power, Levine et al. [132] have applied deep learning to much harder problems of end-to-end robotic manipulation. Our representation of the forward simulation as a differentiable deep network makes risk-aware MPDM (chapter 4 more amenable to learning methods. However, end-to-end learning is non-trivial in our domain. Complex interactions between pedestrians and the robot's closed-loop policies make data collection and generalization challenging.

With its roots tracing back to Stackelberg's game-theoretic modeling [133], bilevel optimization commonly appears in several practical applications such as the toll-setting [42], environmental economics [134], chemical engineering [135] and operations research [136]. Each of these tasks can involve decision making with a hierarchical leader-follower structure where the leader must optimize its decision variables but must account for the followers optimizing their decision variables (pedestrian configurations) in response. In risk-aware MPDM, the robot (leader) must optimize its ego-policy parameters while accounting for the influential pedestrian configuration (follower optimization) in response to the ego-policy.

Unfortunately, bilevel programming is known to be strongly NP-hard [137] and descent based quadratic bilevel programming requires computing the Hessian of the objective function [138] which is computationally infeasible for our application. In order to find effective approximations, we will draw from gradient-based methods first introduced by Arrow, Hurwicz and Uzawa [139]. Even though the analysis of associated saddle-point dynamics is non-trivial [140], gradient dynamics have been widely used in various practical applications such as distributed convex optimization [141], distributed linear programming [142], and power networks [143].

CHAPTER 3

Multi-Policy Decision Making for autonomous navigation in dynamic social environments

3.1 Introduction

In this chapter, we present Multi-Policy Decision Making (MPDM) as a novel approach to motion planning amongst people. Instead of computing a trajectory directly or relying on a single algorithm, in MPDM, a planning process chooses from a set of closed-loop policies by predicting their utilities through forward simulations that capture the coupled interactions between the agents in the environment.

Maneuvering in dynamic social environments is very 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.

POMDPs provide a rigorous formalization for incorporating uncertainty into planning, but rapidly become intractable as the dimensionality of the state-space grows. Recently, approximate POMDP methods based on scenario sampling and forward simulation have been applied to navigation [126] and mapping [127]. Cunningham *et al.* [40] show that, by introducing a number of approximations (in particular, constraining the policy to be one of a finite set of known policies), the POMDP can be solved using MPDM. In their original paper, they use a small set of lane-changing policies; in this chapter, we explore an indoor setting in which the number and the complexity of candidate policies is much higher.

In our setting, dynamic agents (humans) can instantaneously stop or change direction without signaling. We use different and complementary policies than those considered by Cunningham *et al.* [40]: *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 (*Blame*).

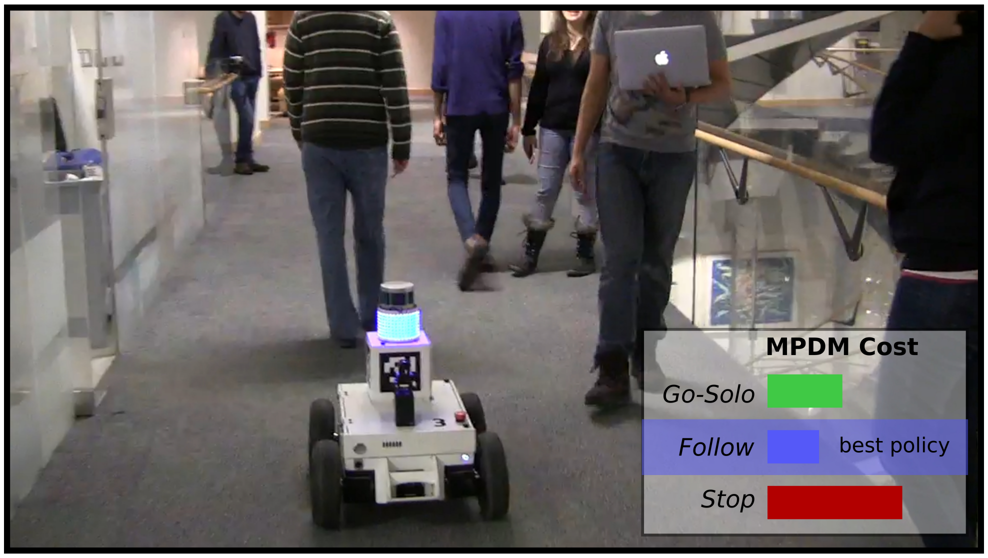


Figure 3.1: Our approach implemented and tested using the MAGIC [144] robot platform. We show that our algorithm is able to navigate successfully on an indoor environment amongst people. MPDM allows the robot to choose between policies. In this case, the robot decides to *Follow* the person in front rather than try to overtake him.

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. 3.1.

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.

3.2 Contributions

1. We show the benefits of switching between multiple policies (as opposed to using a single policy) in terms of navigation performance, quantified by metrics for progress made and inconvenience to fellow agents.

2. We demonstrate the robustness of MPDM to measurement uncertainty and study the effect of the conservatism of the state estimator through simulation experiments (Sec. 3.4.1).

3. Finally, we test the MPDM on a real environment and evaluate the results (Sec. 3.4.2).

3.3 Method

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

Based on past observations, the robot maintains a probabilistic estimate $P(\boldsymbol{x}_0)$ of each observed agent's state - i.e. its position, velocity as well as its inferred policy[1]. An agent's policy $\pi_i = (v_{des}, \boldsymbol{g}_{sub})$, expresses an intent to move towards sub-goal $\boldsymbol{g}_{sub}$ at a desired speed v_{des}. The collective state $\boldsymbol{x}_t \in \mathrm{X}$ consists of the states of the robot and all observed agents at time t. Throughout the paper, we will refer to $\boldsymbol{x}_0$ as the collective state of all agents and the robot's state at the current time. The robot's policy π_r is elected from amongst a set of closed-loop policies Π based on possible outcomes predicted by forward simulating samples drawn from $P(\boldsymbol{x}_0)$.

3.3.1 Candidate Policies

Every agent (including the robot) is assumed to be acting according to some policy. We assume that all agents besides the robot always use the *Go-Solo* policy, but their goals g^i_{sub} and desired future velocities v^i_{des} are ambiguous to the robot. The robot's policy $\pi_r \in \boldsymbol{\Pi}$ is to be selected from a discrete set of closed-loop policies, while the pedestrian policy parameters $(v_{des}, \boldsymbol{g}_{sub})$ are inferred based on past observations and domain knowledge (e.g. salient points in the scene).

We propose the following candidate policy set for navigating amongst pedestrians:

$$\boldsymbol{\Pi} = \{\textit{Go-Solo}, \textit{Follow}_j, \textit{Stop}\}, \tag{3.1}$$

where *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.* [40]. Each policy $\pi_i \in \boldsymbol{\Pi} : \mathrm{X} \mapsto \mathrm{A}_i$. maps a joint state of the system to an action *via a potential field*.

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 $\boldsymbol{a}_i \in \mathrm{A}_i$, is computed using a potential field method that incorporates the effects of obstacles and a goal point based on the Social Force Model as illustrated in Fig. 3.2. This acceleration governs the system dynamics and is determined by the policy π_i followed by the agent as detailed below. The system is constrained to a maximum velocity $|v|_{max}$ for

[1]Several methods can be used for estimating $P(\boldsymbol{x}_0)$ based on past trajectories of agents. We use a Kalman Filter to infer position and velocity and a Naive Bayes Classifier to infer an agent's policy parameters.

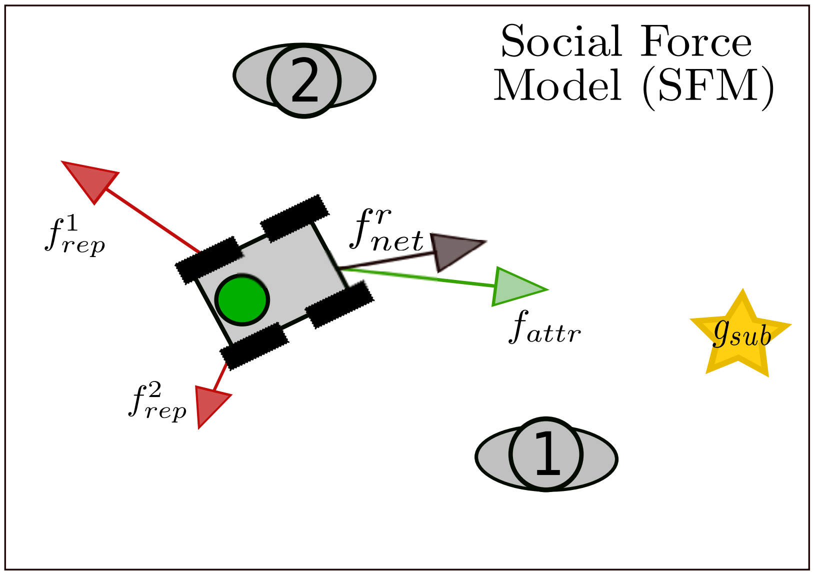


Figure 3.2: An agent's policy $\pi_i = (v_{des}, \boldsymbol{g}_{sub})$, expresses an intent to move towards sub-goal $\boldsymbol{g}_{sub}$ at a desired speed v_{des}. We model policies as reactive potential-field based controllers. At each time-step, an agent i (in this case, the robot) is repelled by other agents (f^j_{rep}) and attracted towards its sub-goal g_{sub} in accordance to the Social Force Model (SFM).

each agent.

Go-Solo

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) [23, 27] to guide it towards its goal. Let $\boldsymbol{e}_{\boldsymbol{p}_i \to \boldsymbol{g}^i_{sub}}$ be the unit vector towards the goal from the agent i. The attractive force acting on the agent is given by:

$$\boldsymbol{f}^i_{attr}(\boldsymbol{x}) = k_{gs} \boldsymbol{e}_{i \to \boldsymbol{g}^i_{sub}}. \tag{3.2}$$

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

$$\boldsymbol{f}^{i,j}_{rep}(\boldsymbol{x}) = a_p e^{-d_{i,j}/b_p} \cdot \boldsymbol{e}_{j \to i}, \tag{3.3}$$

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 [27] .

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 $\{a_o, b_o\}$,

$$\boldsymbol{f}^{i,o}_{obs}(\boldsymbol{x}) = a_o e^{-d_{i,o}/b_o} \cdot \boldsymbol{e}_{o\to i}. \tag{3.4}$$

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

$$\boldsymbol{f}^{i}_{net}(\boldsymbol{x}) = \boldsymbol{f}^{i}_{attr}(\boldsymbol{x}) + \sum_{j\neq i} \boldsymbol{f}^{i,j}_{rep} + \sum_{o\in O} \boldsymbol{f}^{i,o}_{obs} \tag{3.5}$$

The action governing the system propagation is calculated as $\boldsymbol{a}_i = \boldsymbol{f}^{i}_{net}$ (without loss of generality, we assume unit mass). Each pedestrian i tries to maintain its desired speed v^i_{des}. For the robot executing the *Go-Solo* policy, its desired speed v^r_{des} is set to $|v|_{max}$.

Follow-other

In addition to the *Go-Solo* policy, the robot can use the *Follow* policy to deal with certain situations. For example, 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 *Blame*). We propose a reactive *Follow* policy, making minor modifications to the *Go-Solo* policy.

According to the *Follow* policy, the robot *chooses* to follow another agent, the leader, denoted by l. To obtain the resultant force, we can apply the same procedure explained earlier 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

$$\boldsymbol{f}^{r}_{attr}(\boldsymbol{x}) = k_f \boldsymbol{e}_{\boldsymbol{p}_r\to\boldsymbol{p}_l}, \tag{3.6}$$

steers the robot trajectory towards the leader. The other agents and obstacles continue to repel the robot as described in (3.5). Furthermore, the follower tries to maintain the speed of the leader ($v^f_{des} = v_l$).

Stop

The last of the policies available to the robot is the *Stop* policy, where the robot decelerates

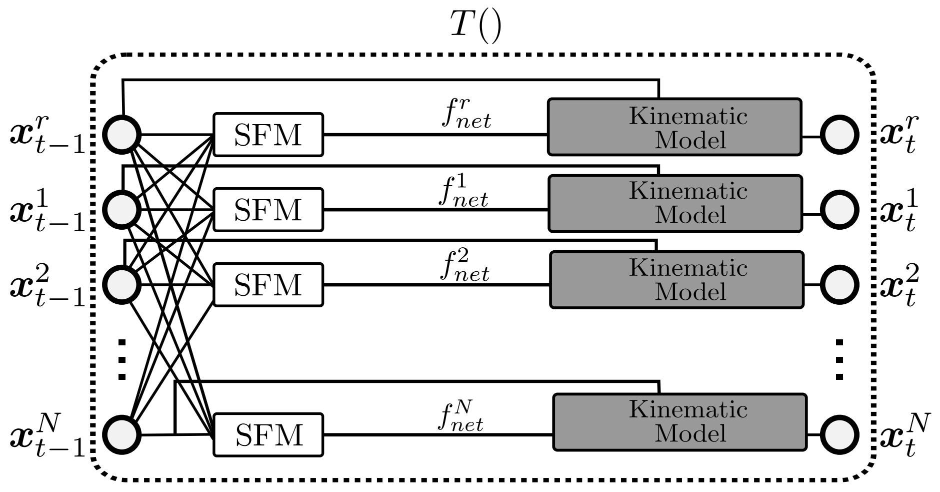


Figure 3.3: Block diagram of the transition function. At each time-step, an agent i (in this case, the robot) is repelled by other agents (f^j_{rep}) and attracted towards its sub-goal g^i_{sub} in accordance to the Social Force Model (SFM). The acceleration $\boldsymbol{a}_i$ is determined by the resultant force f^i_{net} and acts as a control input for the Kinematic Model.

until it comes to a complete stop, according to the following force

$$\boldsymbol{f}^r_{net}(\boldsymbol{x}) = -f_{max}\boldsymbol{e}_{\boldsymbol{v}_r}, \tag{3.7}$$

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

3.3.2 Prediction using Forward Simulation

The transition function $T : \mathrm{X} \mapsto \mathrm{X}.$ maps a given combined state $\boldsymbol{x}_t$ to a new state $\boldsymbol{x}_{t+1}$ through an action $\boldsymbol{a}_t$ (which in our case, is an force) generated by a potential field (Fig. 3.3). The motion of agents is modeled according to a simple double integrator kinematic model in which acceleration, integrated for the length of the time-step, results in a change in velocity. In this way, the transition function $T()$ captures each agent's interactions with all other agents, through the policies they are executing.

We forward simulate the joint initial state $\boldsymbol{x}_0$ until a time horizon H by applying iteratively and simultaneously for each agent. In other words, we recursively apply the transition function $T : \mathrm{X} \to \mathrm{X}$ for H time-steps (through $t = 1, \ldots, H$) to yield a trajectory

$$\begin{aligned}\boldsymbol{X}(\pi_r, \boldsymbol{x}_0) &= \{\boldsymbol{x}_0, T(\boldsymbol{x}_0), T^2(\boldsymbol{x}_0), \ldots, T^H(\boldsymbol{x}_0)\} \\ &= \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_H\},\end{aligned}$$

where $\boldsymbol{x}_t \in \mathrm{X}$ is the collective state consisting of the robot state plus all the agents at time

t of the forward simulation.

This forward roll-out is especially interesting since the pedestrians react to each other as well as with the robot's proposed policy, and vice-versa.

3.3.3 Cost Function

The cost function $C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)$ assigns a scalar value to the outcome of a simulation. We use a cost function that penalizes the inconvenience the robot causes to other agents in the environment (*Blame*) along the predicted trajectory and rewards the robot's progress towards its goal (*Progress*).

Blame: We use the distance to the closest agent as a proxy for the potential disturbance caused to the environment by the robot.

$$B\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big) = \sum_{t=0}^{H} \max_{j \neq r} \mathbf{u}(\|v_r\| - \epsilon) e^{-d_{r,j}(t)/\sigma} \tag{3.8}$$

where $d_{r,j}(t)$ is the distance between the robot and agent j and $\|v_r(t)\|$ is the speed of the robot at time-step t. $\mathbf{u}$ is the step function which is 1 when the argument is ≥ 0 and 0 otherwise. If the robot is in motion ($\|v_r\| > \epsilon$), the decay rate σ determines how much proximity to other agents is penalized.

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

$$PG\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big) = \big(p_r(H) - p_r(0)\big) \cdot \boldsymbol{e}_{p_r \rightarrow g^r_{sub}}, \tag{3.9}$$

where $p_r(H)$ is the position of the robot at end of the forward roll-out and $\boldsymbol{e}_{p_r \rightarrow g^r_{sub}}$ is the unit vector from the current position of the robot to the goal g^r_{sub} .

The resultant cost function is a linear combination of both

$$C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big) = -\alpha PG\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big) + B\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big), \tag{3.10}$$

where α is a weighting factor.

3.3.4 Sampling-Based Multi-Policy Decision Making

In MPDM, the robot dynamically switches from amongst a set of closed-loop policies adapting to different situations.

The policy with the lowest expected cost is chosen and executed until the next planning

Algorithm 1 Multi-Policy Decision Making

```
1: function MPDM(P$\boldsymbol{x}, \boldsymbol{z}, t_H, N_s$)
2:     for $\pi_r \in \Pi$ do
3:         for $s = 1 \dots N_s$ do
4:             $\boldsymbol{x}_0 \sim P(\boldsymbol{x}|\boldsymbol{z})$
5:             $C\big(X(\pi_r, \boldsymbol{x}_0) \leftarrow$ Forward-Simulate($\boldsymbol{x}_0, \pi_r, t_H$)
6:             Score($\pi_r$) $\leftarrow$ Score($\pi_r$) $+ C\big(X(\pi_r, \boldsymbol{x}_0)$
7:         end for
8:         Mean-Score($\pi_r$) $\leftarrow \frac{\text{Score}(\pi_r)}{N_s}$
9:     end for
10:    return $\pi^* = \arg\min_\pi E_{\boldsymbol{x}_0}\{C\big(X(\pi_r, \boldsymbol{x}_0)\big)\}$
11: end function
```

cycle -

$$\pi_r^* = \arg\min_{\pi_r} E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)\}, \tag{3.11}$$

where the cost $C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)$ is associated with the current state $\boldsymbol{x}_0$ upon choosing a policy π.

We used Monte Carlo sampling from the estimator's posterior distribution $P(\boldsymbol{x}_0)$ to approximate the expected cost

$$E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)\} \sim \frac{1}{N}\sum_{n=1}^{N} C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}^n)\big), \tag{3.12}$$

where $\{\boldsymbol{x}^1, \dots, \boldsymbol{x}^N\}$ is the set of samples drawn from the distribution $P(\boldsymbol{x}_0)$. MPDM relies on quick re-planning in order to deal with uncertainty, which constrains the number of forward propagations that can be evaluated per candidate policy π_r.

Note that the cost function C is not only a function of the final state, but also all of the intermediate states through the transition function T_π. Therefore, $C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$ is a highly nonlinear function of robot policy π, and an initial configuration $\boldsymbol{x}_0$ whose evaluation involves a time consuming forward propagation of the system. It might compute, for example, high costs for trajectories that lead to near-collisions.

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.

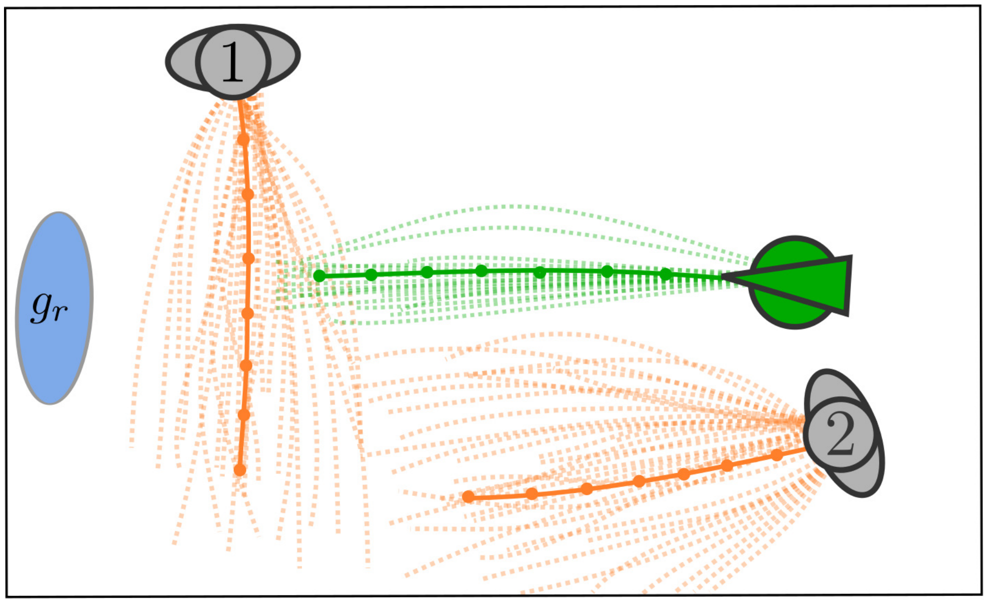


Figure 3.4: Evaluating the *Go-Solo* policy through multiple forward simulations. The Go-Solo policy (where the robot tries to move towards its sub-goal $\boldsymbol{g}_r$ at maximum possible speed) is evaluated by averaging the cost of forward simulations of samples drawn from the estimated distribution of the agents' states (Monte-Carlo sampling). Different sampled initial configurations result in different trajectories. The forward simulations capture closed-loop interaction between the pedestrians and the robot.

3.4 Results

3.4.1 Simulation

We simulate two indoor domains, freely traversed by a set of agents while the robot tries to reach a goal. One simulation '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 (Eqn. 3.12) is set to $N_s = 50$. We use the Intel i7 processor and 8GB RAM for our simulator and LCM [145] for inter-process communication.

Every 300ms (policy election cycle), MPDM chooses a policy. Although the policy election is relatively slow, the robot is very responsive as the policies themselves run at over 50Hz.

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. Our implementation utilizes such

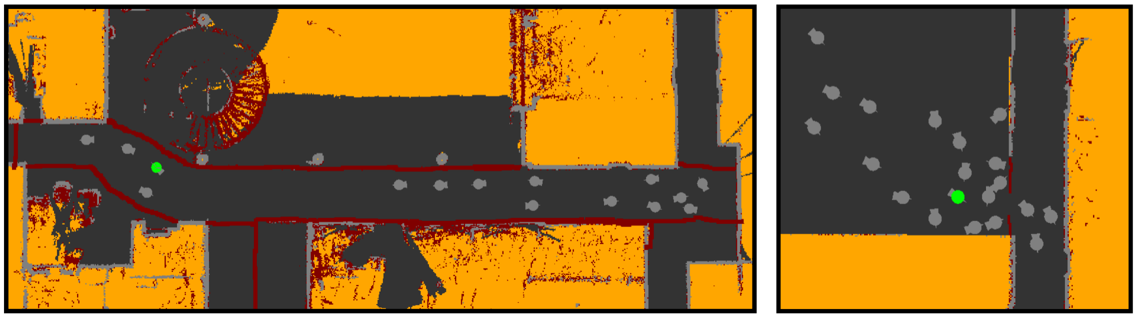


Figure 3.5: The simulated indoor domains chosen to study our approach. *Left*: The hallway domain where 15 agents are let loose with the robot and they patrol the hallway while the robot tries to reach its destination. *Right*: The doorway domain where 15 agents whose goal is reaching the bottom right 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).

domain knowledge through by localizing into a known map, but our approach could be applied more generally.

Domains

The hallway domain (Fig. 3.5-*Left*) is modeled on a $3m \times 25m$ hallway at the University of Michigan. The doorway domain (Fig. 3.5-*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 generated using one hour of simulation data (Fig. 3.6-*Top*), 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 $1.8m/s$. 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 $4s$ into the future.

Empirical Validation

Fig. 3.6 demonstrates the benefits of MPDM through an example simulation run. 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 *Blame* no longer inconveniencing other agents.

This observation is strengthened by the empirical distributions of the metrics generated

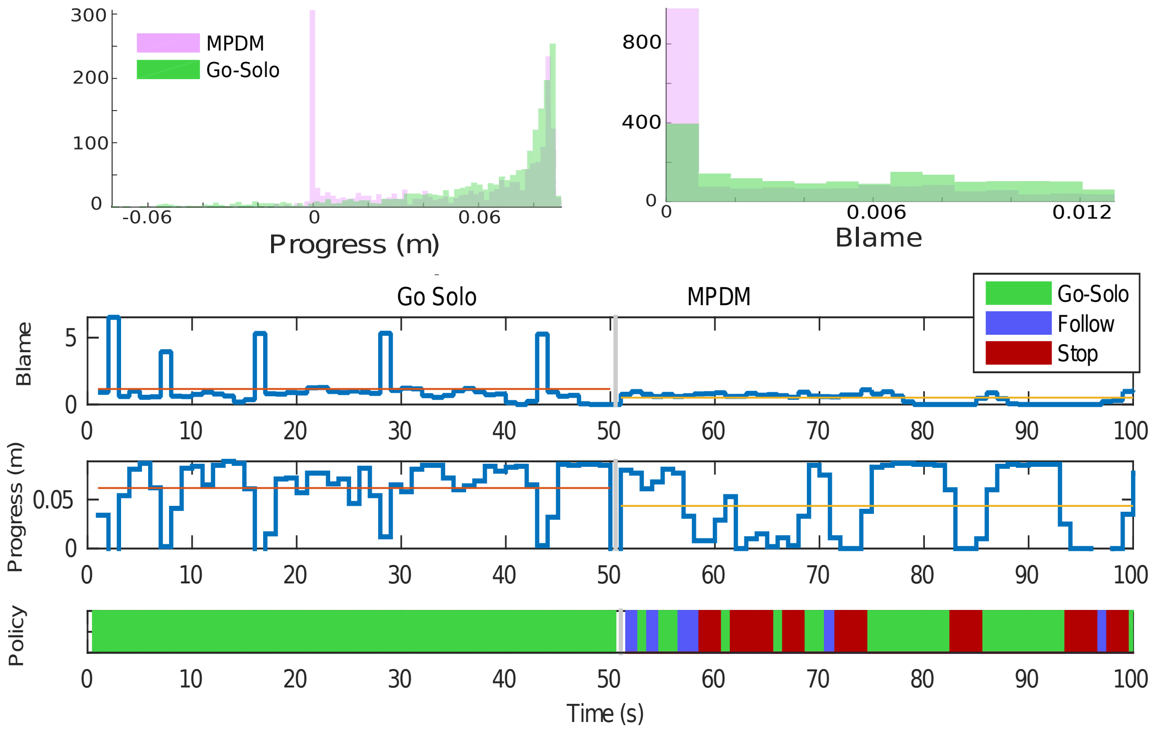


Figure 3.6: Qualitative comparison of MPDM and the exclusive use of *Go-Solo* in a simulated environment. *Top*: Distributions of the evaluation metrics - *Blame* and *Progress* generated using one hour of simulated data. Ideal behavior would give rise to high *Progress* and low *Blame*. The higher valued mode for *Blame* denote undesirable behavior (close encounters), which MPDM is able to avoid. *Bottom*: 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 *Blame*, manifesting as undesired peaks. In the next $50s$, when the robot executes MPDM, the *Blame* curve is almost flat, indicating that nearby interactions are reduced drastically.

from 30k samples. We notice that the *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.

Experiments with Observation 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.

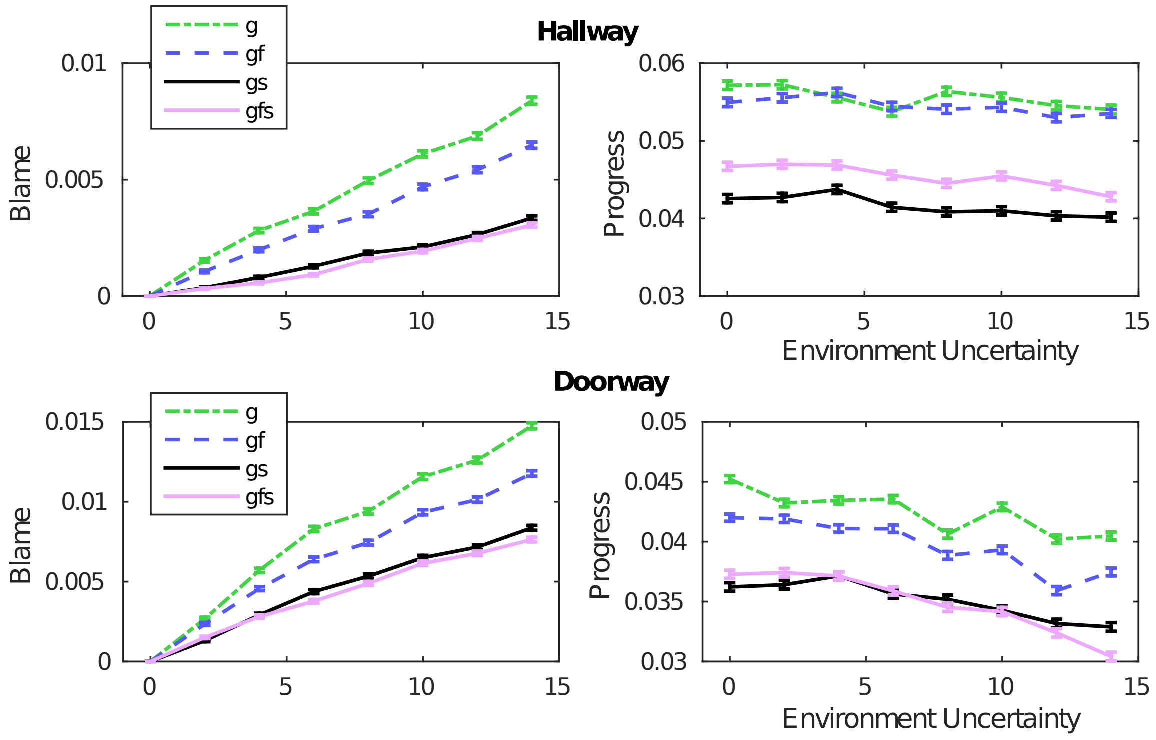


Figure 3.7: Simulation results varying uncertainty in the environment (k_z) for a fixed posterior uncertainty (k_e). We show results for 4 combinations of the policies, varying the flexibility of MPDM: *Go-Solo* (g), *Go-Solo* and *Follow* (gf), *Go-Solo* and *Stop* (gs) and the full policy set (gfs). The data collected from 100 epochs (30k samples) is averaged in groups of 10. We show the mean and standard error. Ideally, we would want low *Blame* and high *Progress*. *Left:* A lower *Blame* indicates better behavior as the robot is less often the cause of inconvenience. Increasing the noise in the environment makes the robot more susceptible to disturbing other agents and vice-versa. We can observe that the *Blame* 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 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 *Blame* as compared to (gf).

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 $\mathrm{diag}(\sigma_{p_x}, \sigma_{p_y}, \sigma_{|v|}, \sigma_\theta)$. We do not perturb the goal. These uncertainties are propagated during the estimation of the posterior state $P(\boldsymbol{x}|\boldsymbol{z})$.

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.

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

1. varying k_z for a fixed k_e to understand how MPDM performs when varying uncertainty in the environment (Fig. 3.7).

2. varying $\frac{k_e}{k_z}$ to understand how MPDM performs when the robot's estimator overestimates/underestimates the uncertainty in the environment (Fig. 3.8).

First, we 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. 3.7 shows the performance of the robot for the hallway and the doorway domain respectively. We observe that the *Blame* increases at the 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* lower.

Next, we 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 ratio, we average over the values of k_z. Fig. 3.8 shows performance trends for both the domains. 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

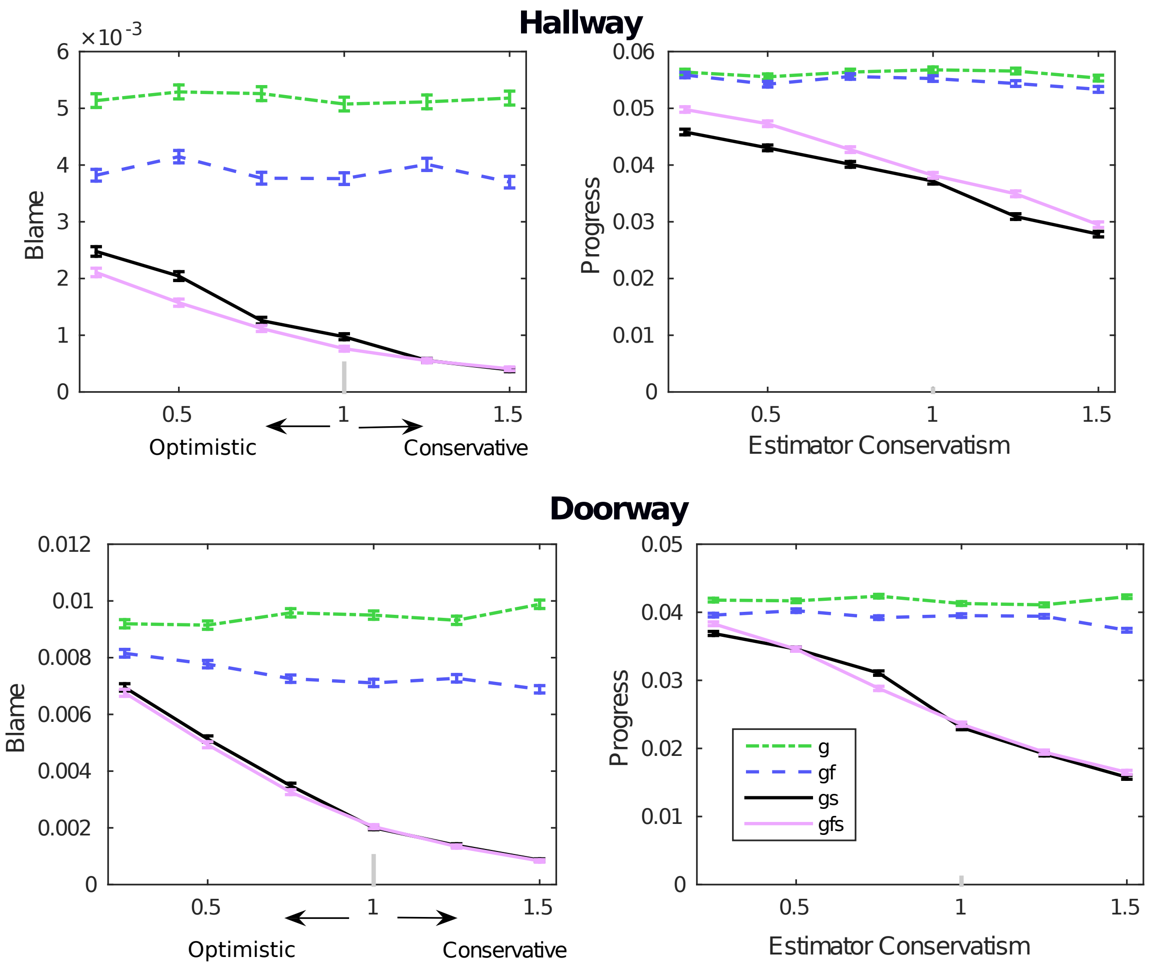


Figure 3.8: Navigation performance varying the degree of conservatism ($\frac{k_e}{k_z} = \{0.25, 0.5 \dots 1.5\}$) of the estimator averaged over $k_z = \{2, 4, \dots, 14\}$. Combinations of the policies as presented in Fig. 3.7. The data collected from 100 epochs (30k samples) is averaged in groups of 10. We show 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.

Progress and *Blame* increases. Even in these situations, the flexibility of multiple policies enables navigation with lower *Blame*.

3.4.2 Real-World Experiments

Our real-world experiments have been carried out in the hallway that the simulated hallway domain (Sec. 3.4.1) was modeled on. We implemented our system on the MAGIC robot [144], a differential drive platform equipped with a Velodyne VLP-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. 3.4.1. 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. 3.9 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*. As soon as the dynamic obstacles are no longer a hindrance, the robot changes the policy to execute and *Goes-Solo*. In Fig. 3.9-*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[2] clearly shows the limitations of the *Go-Solo* and how MPDM solves these limitations.

Fig. 3.10 shows the results of our robot executing MPDM as compared to a constant navigation policy - *Go-Solo*. As discussed before in Sec. 3.4.1, we show that our observations based on simulations hold in real environments. Specifically, MPDM performs much better, roughly 50%, in terms of *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.

3.5 Summary

This chapter described a novel approach for autonomous navigation among uncertain dynamic agents by choosing from amongst a set of closed-loop policies - {*Go-Solo, Follow,*

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

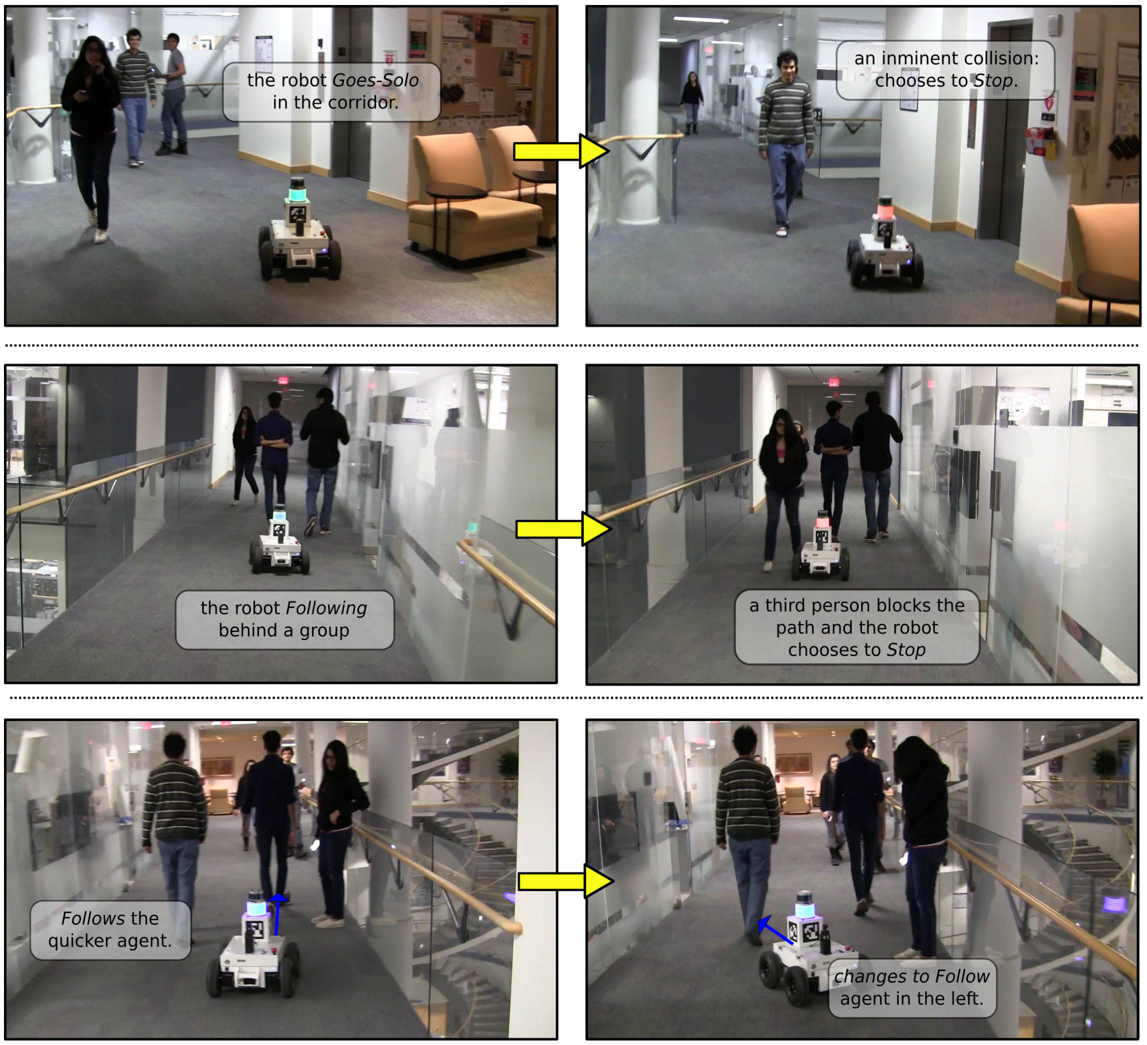

the robot *Goes-Solo* in the corridor.
an inminent collision: chooses to *Stop*.
the robot *Following* behind a group
a third person blocks the path and the robot chooses to *Stop*
Follows the quicker agent.
changes to Follow agent in the left.

Figure 3.9: Real situations illustrating the emergent behavior from MPDM. The left column depicts three scenarios (each row has a different scenario) while testing the robot navigation in a real environment. The right column shows 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.

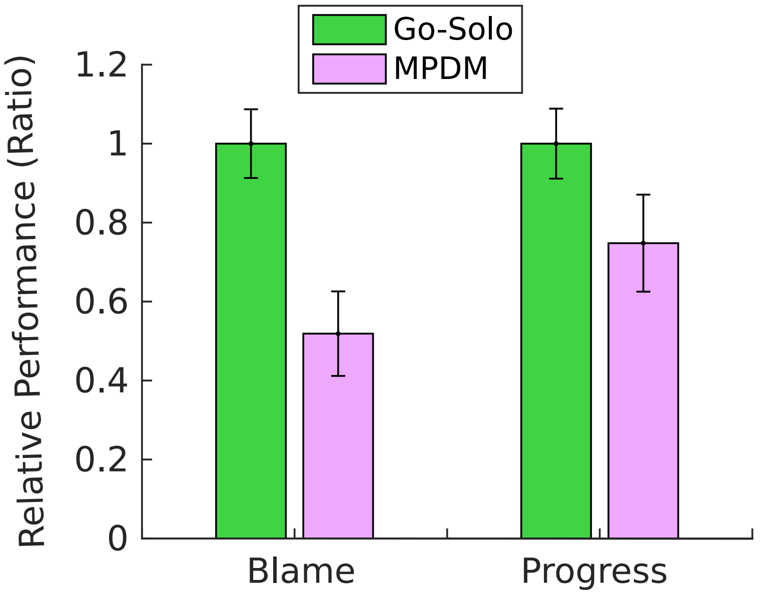


Figure 3.10: The mean and standard error for the performance metrics over 10 second intervals (groups of 10 samples) using data from 40 minutes of real world experiments. All measures are normalized based on the corresponding mean value for the *Go-Solo* policy. This figure demonstrates that our results obtained in simulations (Sec. 3.4.1) hold on real environments. MPDM shows much better *Blame* costs than only *Go-Solo* at the price of slightly reducing its *Progress*.

Stop} to adapt to different situations. Each candidate policy was evaluated based on forward simulations of samples drawn from the estimated distribution of the agents' states (Monte-Carlo sampling). These forward simulations and thereby the cost function, capture agent-agent interactions as well as agent-robot interactions which depend on the policy being evaluated. The robot is responsive to sudden changes in the environment due to quick replanning (every 300ms).

CHAPTER 4

Risk-Aware Multi-Policy Decision Making

4.1 Introduction

In the Multi-Policy Decision Making (MPDM) framework, every 300ms, a robot's policy is elected by sampling from the distribution of current states, predicting future outcomes through forward simulation, and selecting the policy with the best expected performance. More the samples used to approximate the expected utility, better is the reliability of the estimate; however, forward-simulations are computationally intensive, and MPDM must evaluate a policy in relatively few (about 50) forward simulations. In cases where the forward simulated trajectories are more sensitive to initial configurations, and in environments with a large number of possible outcomes, policy evaluation becomes more challenging (Fig. 4.1). In this chapter, we address the problem of reliably evaluating as few forward simulations as possible.

Based on past observations, the robot not only maintains the estimate of the current position and velocity for each pedestrian, but also their future intentions (policies). The dimensionality of the space of all possible initial configurations is very large. The key challenge arises from the uncertainty associated with the inferred state of the agents (people) and the complex multi-agent interactions which make the forward-simulated trajectories sensitive to the initial configurations sampled. In our application, it is especially difficult to sample bad outcomes because we assume that all agents follow policies that tend to avoid collisions and dangerous scenarios in the first place. The bad outcomes arise from a few initial configurations whose neighborhoods may be fairly uninteresting (Fig. 4.1). Collisions are profoundly serious, but near misses are mundane. In other words, the cost function tends to be tightly peaked around these configurations. This problem is more prevalent in complex multi-agent scenarios.

Sampling randomly is likely to miss high-cost events, even if they are individually reasonably probable (high probability density) because of the scarcity of such configurations

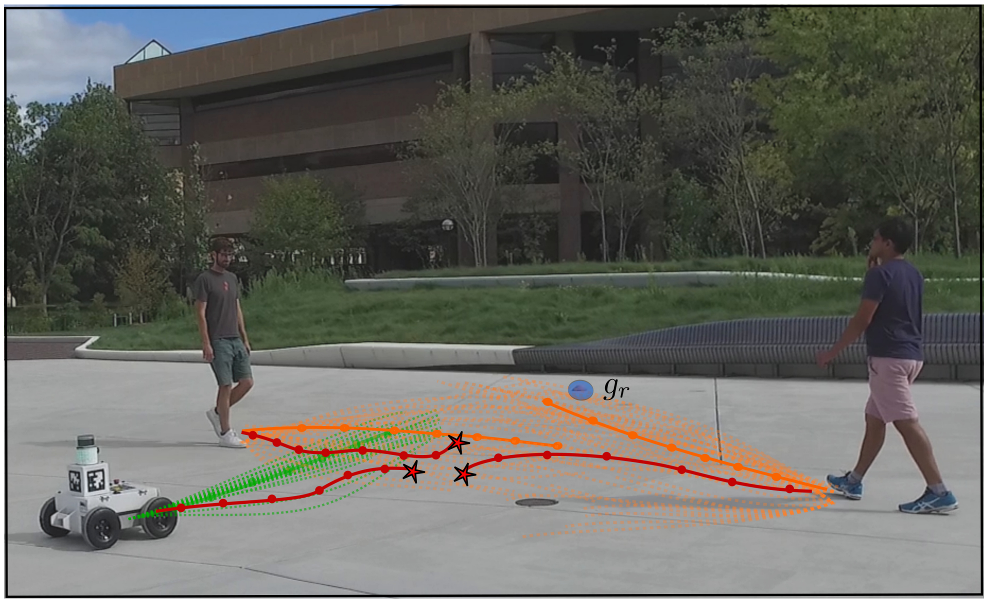


Figure 4.1: An illustration motivating risk-aware planning. The trajectories arising from the most likely initial configuration for the agents (orange) and the robot (green), like most outcomes of possible initial configurations (dashed lines) are benign. Near misses are mundane as the agents in the forward simulation tend to avoid collision. As a result, there may be a few dangerous initial configurations that may be individually likely and yield high-cost outcomes (red). While evaluating policies, likely collisions (red stars) should be discovered as quickly as possible to allow larger candidate policy sets for MPDM.

in the state space (low total probability mass of high-cost outcomes). Discovering these configurations through random sampling may require drawing many samples, which becomes a performance bottleneck. Without enough samples to find influential outcomes, the quality of planning suffers. Addressing this issue is crucial for reliable systems in applications such as autonomous cars, navigation in social environments, etc.

The key insight is that we need to explicitly search for influential outcomes (those that have high cost and probability) because they influence our decision making process the most. Rather than random sampling, by biasing the sampling of initial configurations towards likely, high-cost outcomes, we show how policies can be evaluated efficiently in a risk-aware fashion.

4.2 Contributions

1) We formulate a risk-aware objective for evaluating policies to bias the sampling of initial configurations towards likely, high-cost outcomes. This formulation casts the problem of policy evaluation as an optimization problem.

2) Next, we show that our forward simulation can be encoded in a differentiable deep network and propose an efficient procedure using backpropagation to compute an accurate gradient of the objective function without the need for heuristics. Our optimization algorithm is anytime, finding increasingly influential configurations at every iteration.

3) We demonstrate the efficiency of this algorithm through extensive simulation experiments and show that using only 50 samples, our proposed method can perform comparably to sampling with 500 samples.

4) Finally, we incorporate this idea into MPDM and demonstrate significant performance improvements on a real robot platform navigating in a semi-crowded highly dynamic environment.

4.3 Backprop-MPDM

Previous formulations of MPDM used Monte Carlo sampling from the estimator's posterior distribution $P(\boldsymbol{x}_0)$ to approximate the expected cost

$$E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\} \sim \frac{1}{N}\sum_{n=1}^{N} C\big(\boldsymbol{X}(\pi,\boldsymbol{x}^n)\big), \tag{4.1}$$

where $\{\boldsymbol{x}^1,\ldots,\boldsymbol{x}^N\}$ is the set of samples drawn from the distribution $P(\boldsymbol{x}_0)$.

However, Monte-Carlo sampling over the posterior $P(\boldsymbol{x}_0)$ often fails to capture miss high-cost events that should guide decision making because of the sparsity of such configurations in the state space (low total probability mass of high-cost outcomes) as illustrated in Fig. 4.1.

Importance Sampling

Importance sampling is used to simulate rare events that are often missed by Monte-Carlo sampling [146]. In our domain, rather than sampling pedestrian configurations from $P(\boldsymbol{x}_0)$, a *proposal* distribution $Q(\boldsymbol{x}_0)$ can used to bias sampling towards high-cost outcomes. However, finding a good proposal distribution in our domain is challenging because the agent-agent interactions make the predicted trajectories and the associated cost function sensitive to the pedestrian configuration $\boldsymbol{x}_0$. The dimensionality of the space of all possible initial configurations is very large and importance sampling with only a few samples can result in a high-variance or a high-bias estimate of the expected cost, depending on the choice of the proposal distribution.

The *maximum principle* prescribes sampling from a proposal distribution with its mode at $\arg\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\}$ in order to capture high-risk outcomes [147] and is a widely used in the field of quantitative risk-management. A multi-variate Gaussian centered at $\arg\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\}$ is a popular choice to obtain an unbiased estimate of the expected cost. However, since our cost function may be multi-modal and 'spiky', the importance weights $\frac{P(\boldsymbol{x}_0)}{Q(\boldsymbol{x}_0)}$ for samples at the tails of $Q(\boldsymbol{x}_0)$ can be very large, resulting in high-variance estimates when there are a small number of samples. Instead, if $Q(\boldsymbol{x}_0)$ is a truncated Gaussian, we get lower variance, but this estimator is now biased towards likely high cost initial configurations because truncated distribution's support is a subset of the posterior distribution's support.

Policy Evaluation as an optimization problem

Rather than try to approximate the expected utlity with a few samples, we explicitly search for influential outcomes (those that have high cost and probability) because they influence our decision making process the most. We propose a new risk-aware objective for evaluating policies inspired by the maximum principle –

$$\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\}. \tag{4.2}$$

This objective allows us to use optimization techniques; unlike sampling, it does not matter *how* we find the maximizing configuration. We can perturb the state elements of $\boldsymbol{x}_0$ while sampling in order to find high $P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)$ outcomes. However, the samples used to search are no longer random samples from the posterior distribution and cannot be used to approximate the expectation according to Eq. 4.1.

One could set aside some samples for the optimization while the remaining samples could be used to calculate a more informed estimate of expectation by constructing a proposal distribution around the likely and dangerous configurations (importance sampling). In this way, our method can be used to augment sampling. We do not explore this option in our experiments. Instead, we change our decision making rule as follows

$$\pi^* = \arg\min_{\pi}\Big[\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\}\Big]. \tag{4.3}$$

Naturally, changing the objective function for policy evaluation from Eq. 3.12 to Eq. 4.2 changes the emergent behavior of the planner, and the design of the cost function may need to be adjusted. In practice, though, we have had no difficulty adapting our systems to the proposed objective function. For safety-critical systems in which it makes sense to be risk-aware, using Eq. 4.3 may even be a more natural choice than minimizing the expected cost of an outcome.

Even with the new objective, the same challenges remain - we need to quickly discover potentially dangerous outcomes when the dimensionality of the search space scales linearly with the number of pedestrians (5 pedestrians would yield a 20-dimensional search space) and the complex multi-agent interactions make the forward-simulated trajectories sensitive to the initial configurations. In the remainder of this section, we present an anytime algorithm for discovering influential configurations.

4.3.1 Network Architecture

Deep neural networks model complex functions by composing (chaining) relatively simple functions (convolutions or ReLU modules). Similarly, a forward simulation captures the complex dynamics of the system using simple one-step transition functions T.

Since our cost function is a linear combination of costs computed along the trajectory, we can conceptualize the forward simulation as a deep network (Fig. 5.2) that outputs a trajectory cost $C(\boldsymbol{X}(\boldsymbol{x}_0))$ based on the input initial configuration $\boldsymbol{x}_0$.

Let $L_t(\boldsymbol{x}_t)$ be the cost accrued at time-step t for the state $\boldsymbol{x}_t$. We define a function $\Phi(t, \boldsymbol{X})$ that accumulates the cost of a trajectory, from the final time H backwards to the initial time $t = 0$

$$\Phi(t, \boldsymbol{X}) = \sum_{\tau=t}^{H} L_\tau(\boldsymbol{x}_\tau). \tag{4.4}$$

Our objective cost can be expressed as $C(\boldsymbol{X}) = \Phi(0, \boldsymbol{X})$. We can formulate Φ recursively as:

$$\Phi(t, \boldsymbol{X}) = \Phi(t+1, \boldsymbol{X}) + L_t(\boldsymbol{x}_t). \tag{4.5}$$

We want to compute $\nabla_{\boldsymbol{x}_0} C(\boldsymbol{X}) = \nabla_{\boldsymbol{x}_0} \Phi(0, \boldsymbol{X})$. The gradient of the cost at time-step H is

$$\nabla_{\boldsymbol{x}_H} \Phi(H, \boldsymbol{X}) = \frac{\partial \Phi(H, \boldsymbol{X})}{\partial \boldsymbol{x}_H} = \frac{\partial L_H(\boldsymbol{x}_H)}{\partial \boldsymbol{x}_H}. \tag{4.6}$$

We can compute the gradient iteratively from time-step H backwards to $t = 0$ by

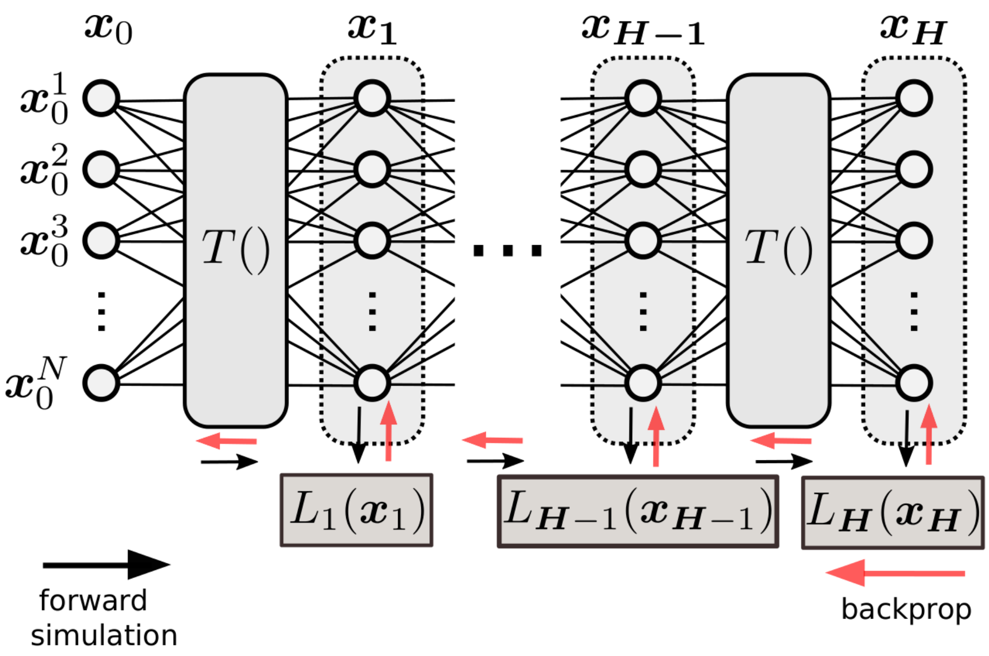


Figure 4.2: A deep network representation for our cost function. The initial configuration $\boldsymbol{x}_0$ propagates through several layers, each representing the transition function T. The output of layer t determines a cost $L_t(\boldsymbol{x}_t)$. Our cost function $C(\boldsymbol{X}(\boldsymbol{x}_0))$ accumulates costs calculated at each time-step along the forward simulated trajectory.

applying (4.5) and expanding terms:

$$\begin{aligned}
\nabla_{\boldsymbol{x}_t}\Phi(t,\boldsymbol{X}) =& \frac{\partial\Phi(t,\boldsymbol{X})}{\partial\boldsymbol{x}_t} = \frac{\partial\{\Phi(t+1,\boldsymbol{X}) + L_t(\boldsymbol{x}_t)\}}{\partial\boldsymbol{x}_t} \\
&= \frac{\partial\Phi(t+1,\boldsymbol{X})}{\partial\boldsymbol{x}_t} + \frac{\partial L_t(\boldsymbol{x}_t)}{\partial\boldsymbol{x}_t} \\
&= \frac{\partial\Phi(t+1,\boldsymbol{X})}{\partial\boldsymbol{x}_{t+1}}\frac{\partial\boldsymbol{x}_{t+1}}{\partial\boldsymbol{x}_t} + \frac{\partial L_t(\boldsymbol{x}_t)}{\partial\boldsymbol{x}_t} \\
&= \frac{\partial\Phi(t+1,\boldsymbol{X})}{\partial\boldsymbol{x}_{t+1}}\boxed{\frac{\partial T(\boldsymbol{x}_t)}{\partial\boldsymbol{x}_t}} + \frac{\partial L_t(\boldsymbol{x}_t)}{\partial\boldsymbol{x}_t}. \qquad (4.7)
\end{aligned}$$

Eqn. 5.4 can be used to efficiently compute $\nabla_{\boldsymbol{x}_0}C(\boldsymbol{X})$ as long as the gradient of transition function can be computed effectively using backpropagation (Fig. 5.2).

4.3.2 Enabling Effective Backpropagation

We have recognized that the kinematic models used for the agents have an impact on the quality of the gradients. Our previous implementation of MPDM (Alg. 1) used a simple double integrator model for all agents with heuristics to restrict lateral motion for more realistic simulation. While the simple model was useful for fast forward simulation, the

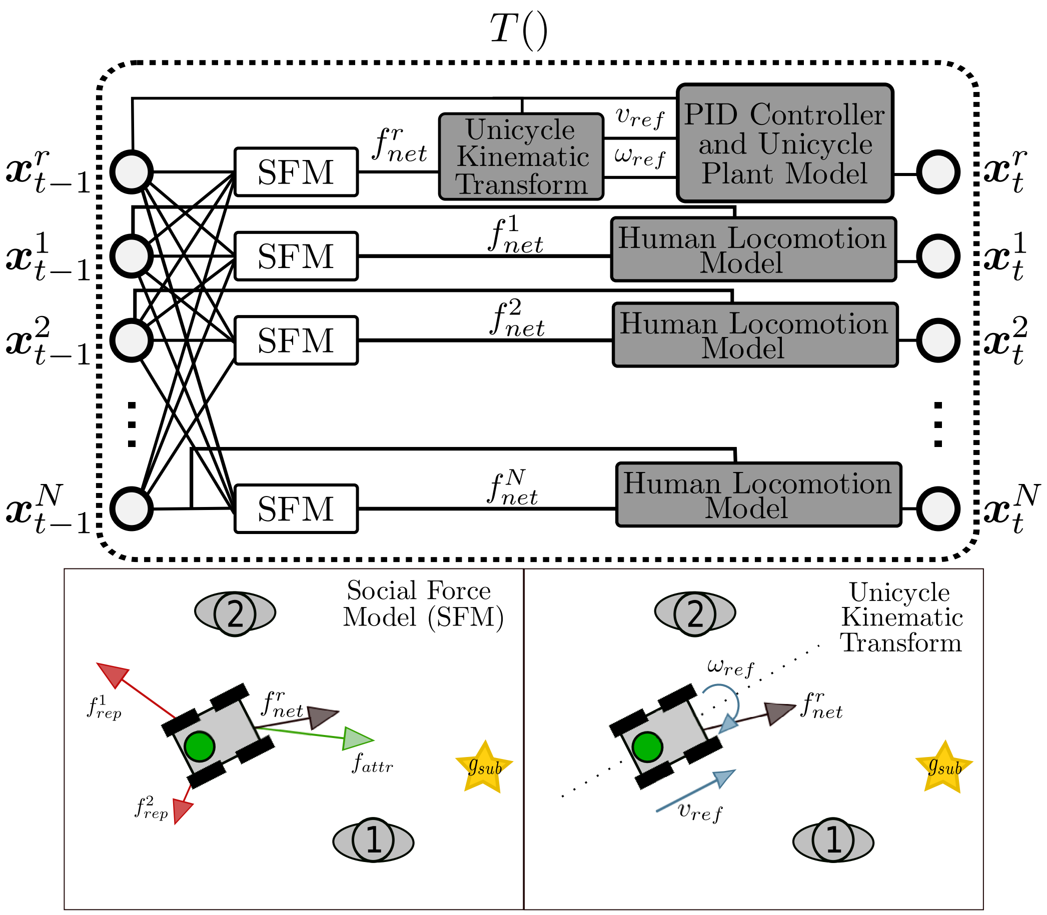


Figure 4.3: Block diagram of the transition function. At each time-step, an agent i (in this case, the robot) is repelled by other agents (f^j_{rep}) and attracted towards its sub-goal g_{sub} in accordance to the Social Force Model (SFM). Pedestrians are modeled using the HSFM model where the social force [66] acts as a control input for the Human Locomotion Model. The robot is modeled like a unicycle and the social force f^r_{net} is transformed into a compliant reference signal (v_{ref}, ω_{ref}) for a lower-level velocity controller.

heuristics contain hard thresholds that manifest as zeros in the matrix $\frac{\partial T(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}$. As a result, useful gradients are truncated (as highlighted by the box in Eqn. 5.4 hampering effective backpropagation.

Here, we use non-holonomic kinematic models that augment the agent's state with angular velocity to capture the effect of lateral forces. This model ensures the differentiability of T while maintaining realistic human motion in the forward simulation.

Specifically, we use the headed social force model (HSFM) [66] for all the pedestrians and a unicycle-like model for the robot as described below. For the robot, the net force is computed using the SFM f^r_{net}, but due to the inherent constraints on a wheeled platform, we transform f^r_{net} into a compliant reference signal (v_{ref}, ω_{ref}) for a lower-level velocity

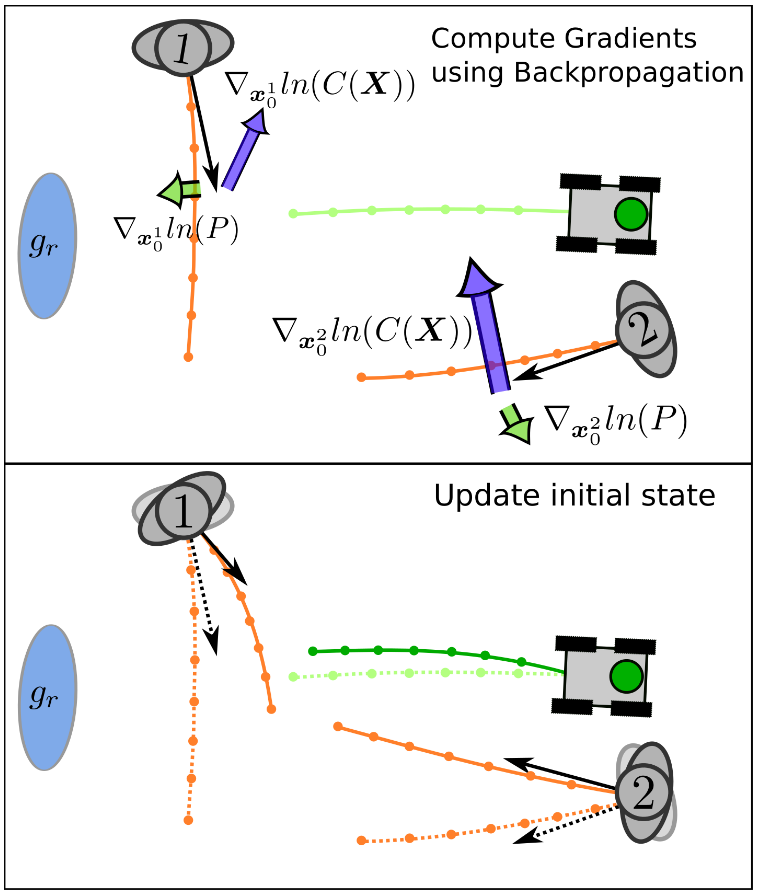

Compute Gradients using Backpropagation
Update initial state
g_r
$\nabla_{\boldsymbol{x}_0^1} ln(C(\boldsymbol{X}))$
$\nabla_{\boldsymbol{x}_0^1} ln(P)$
$\nabla_{\boldsymbol{x}_0^2} ln(C(\boldsymbol{X}))$
$\nabla_{\boldsymbol{x}_0^2} ln(P)$
1
2

Figure 4.4: Backpropagation finds increasingly influential outcomes. The forward propagated outcome of the sampled initial configuration (*Top*) is not discouraging for the robot as it does not inconvenience either agent. For agents $i = \{1, 2\}$, the computed gradients $\nabla_{\boldsymbol{x}_0^i} ln\big(C(\boldsymbol{X})\big)$ (Blue) drive the agents towards configurations where the robot would inconvenience them under its current policy while $\nabla_{\boldsymbol{x}_0^i} ln\big(P(\boldsymbol{x}_0)\big)$ (Green) drive them to more likely configurations. The agents can be simultaneously updated resulting in a more influential configuration (*Bottom*).

controller

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix}_{t+1} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} f^r_{net}. \tag{4.8}$$

The lookahead distance l determines the tendency of the robot to turn in order to compensate for the lateral force. The robot's state is then propagated towards the reference signal using a first-order model for each of the independent wheel velocity controllers and a unicycle plant model.

Our proposed transition function layer $T(\boldsymbol{x}_t)$ (Fig. 4.3) allows us to compute accurate gradients of the transition function. Eqn. 5.4 can now be implemented efficiently via backpropagation, where $\frac{\partial T(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}$ and $\frac{\partial L_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}$ are computed during the forward propagation and cached.

Fig. 5.3 illustrates one iteration of gradient descent using backpropagation through a simple initial configuration $\boldsymbol{x}_0$ consisting of two agents and the robot executing the *Go-Solo* policy towards its goal g_r.

Algorithm 2 describes the policy election for risk-aware MPDM. Provided with a probability distribution over initial configurations, $P(\boldsymbol{x}_0)$, a set of candidate policies, Π, and a forward simulation budget, N_π, each candidate policy is evaluated (scored) according to the most influential (worst-case) outcome discovered within the computational budget.

The objective function $P(\boldsymbol{x}_0)C(\boldsymbol{X})$ can have multiple local-minima depending on the number of agents and the complexity of the initial configuration. Finding the global maximum through exhaustive search is computationally infeasible due to the large state-space. Our goal is to quickly find an influential configuration whose value is comparable to the global optimum even if it may not be the highest-valued configuration.

Algorithm 2 Policy Election for Risk-aware MPDM

```
 1: function POLICY-ELECTION LOOP($P(\boldsymbol{x}), \Pi, N_\pi$)
 2:     for $\pi \in \Pi$ do
 3:         Initialize $U_\pi, n \leftarrow 0$
 4:         while $n < N_\pi$ do
 5:             Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x})$
 6:             $\boxed{U^*, n_{opt} \leftarrow \text{Optimize}(\boldsymbol{x}_0, \pi)}$
 7:             $n \leftarrow n + n_{opt}$
 8:             $U_\pi \leftarrow \max\{U^*, U_\pi\}$
 9:         end while
10:     end for
11:     $\pi^* \leftarrow \arg\min_\pi U_\pi$
12: end function
```

Our algorithm samples an initial configuration from $P(\boldsymbol{x}_0)$ (Line 5) and optimizes it, perturbing the sampled configuration iteratively towards increasingly influential outcomes until convergence to a local optima whose objective function value is U^* (Line 6). The number of forward simulations n_{opt} used by an optimization procedure corresponds to its rate of convergence. Upon convergence, a new initial configuration is sampled and this process is repeated until the forward simulation budget N_π is consumed. The utility of a policy U_π is the most influential (highest-valued) configuration encountered. The policy with the least risk is elected.

Our heuristic-based stochastic gradient method (SGA) [148] computed approximate gradients for each agent and perturbed one agent at a time to avoid divergence. In contrast, by computing accurate gradients, Backprop-MPDM (BP) can perturb all the agents simultaneously without divergence. The gradient also accounts for agent-agent interactions as well as static obstacles. Our experimental results demonstrate the benefits of Backprop-MPDM for optimizing the probabilistic cost surface (Line 6).

4.4 Results

Our operating environment is an open space, freely traversed by a set of agents while the robot tries to reach a goal. The unconstrained nature of this domain makes the trajectories more dependent on initial configurations. Agents can randomly slow down or come to a stop. We set to $\alpha = 5$ using a procedure similar to [29] so that both *Blame* and *Progress* have more or less equal impact on the cost function. One simulation 'epoch' consists of a random initialization of agent states followed by a 5 minute simulated run at a granularity $\Delta t = 0.1$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 $4s$ into the future.

4.4.1 Efficiency of Search

First, we study the efficiency of different search strategies in approximating our objective function - $\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)\}$.

We generated a dataset consisting of 16k randomly chosen simulated scenarios where at least one agent was present within 5m of the robot. We then sort them based on the number of agents in the robot's neighborhood. Our objective function is defined over innumerable possible initial configurations belonging to a high-dimensional continuous space that scales linearly with the number of agents considered. For each scenario, 2k random samples were optimized and the worst-case outcome was used to approximate the global optimum.

We now vary the number of agents in the robot's vicinity, thus increasing the complexity of the scenario and the dimensionality of the state space. For reliable real-time policy evaluation, influential outcomes must be detected quickly. We estimate the number of iterations needed by each algorithm to achieve a certain fraction (50%) of the worst outcome in the dataset (find an influential outcome).

For each algorithm, the experiment is run 1k times on each scenario. We use bootstrap sampling (with replacement) on our data-set as in [148] to estimate the mean and standard error of their performance.

Stochastic Gradient Ascent [148] computes approximate agent-specific gradients of a simplified cost function. In order to limit the divergence arising due to these approxi-

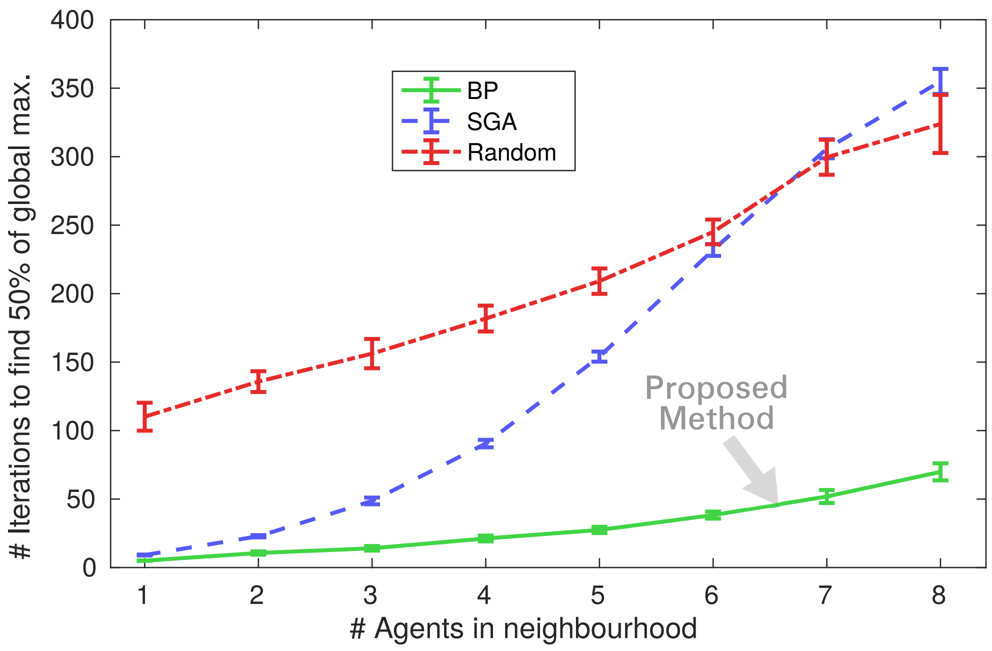


Figure 4.5: Degradation of Stochastic Gradient Ascent (SGA) [148] in crowded scenarios. For each algorithm, we estimate the mean and standard error of the number of iterations (forward simulations) taken to discover an influential outcome varying the number of agents in the robot's vicinity, and thereby the dimensionality of the search space. The lower the slope, the better, more robust the algorithm to complex scenarios with high-dimensional search spaces. Random sampling, as expected, requires many samples even in simpler configurations (1 agent). SGA cannot find influential outcomes efficiently in complex scenarios with multiple agents, scaling so poorly that for more than 6 agents it performs worse than random sampling. Backprop-MPDM (BP) is able to find those adverse outcomes even for crowded scenarios with 8 people.

mations, the stochastic gradients are ranked using a heuristic function and only the most promising agent is perturbed at a time. Despite performing well in scenarios involving few agents, this method does not scale well to more challenging crowded settings. Fig. 4.5 shows that although all the algorithms take longer to find influential outcomes as the complexity of the environment grows, the performance of SGA deteriorates sharply for more than 3 agents. Beyond 6 agents, it performs as poorly as random sampling since it takes a long time to converge from a sampled initial configuration to a local optimum. Backpropagation, on the other hand, overcomes these limitations as it computes accurate gradients, and all agents can simultaneously be updated without divergence.

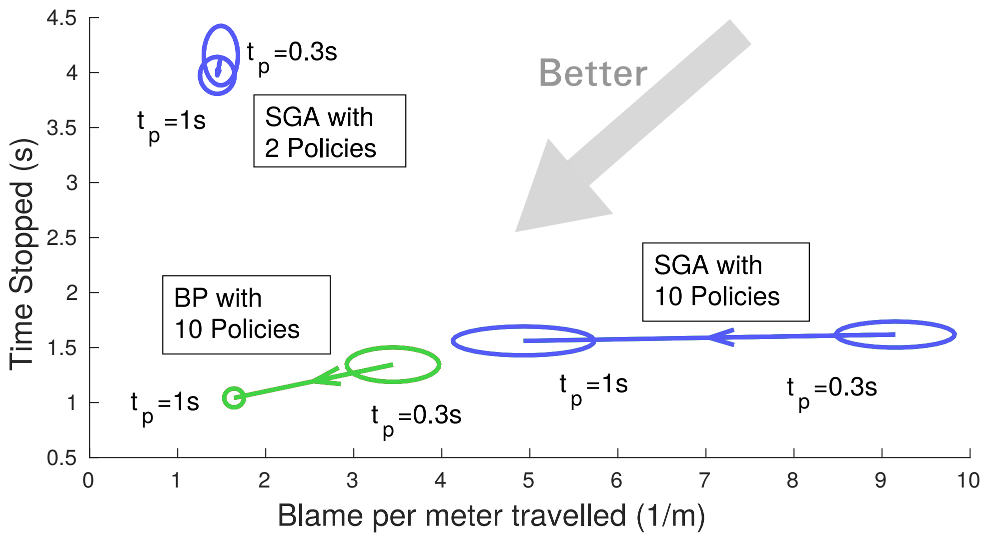


Figure 4.6: Our proposed method, Backprop-MPDM (BP) can evaluate 10 policies reliably in real-time, while SGA cannot. We compare the performance of various algorithms on 6 hours of navigation in our simulated environment. We measure the *Time Stopped* for every goal reached as well as the *Blame per meter traveled* by the robot. For each algorithm, we use bootstrap sampling to estimate the mean and standard error for these metrics, represented by the axes of an ellipse. Lower the *Blame* or *Time Stopped*, the better. We run the simulator in real-time allowing a planning time $t_p = 0.3s$. Although SGA can evaluate the smaller policy set reliably in real-time, the lack of options results in frequent *Stopping*. Unfortunately, SGA cannot evaluate a larger policy set of 10 policies reliably and accumulates large *Blame*. Since BP can evaluate the larger policy set more quickly and reliably than SGA , the robot navigates safely (low *Blame*) in real-time without *Stopping* unnecessarily. The benefits of risk-aware MPDM with the larger policy set can also be observed in the attached video. Upon slowing down the simulator (three times slower than real-time) to allow an unrealistic planning time of $t_p = 1s$, we observe that SGA with 10 policies is able to drastically reduce *Blame*. However, even then BP outperforms SGA.

4.4.2 Increasing the Number of Candidate Policies

Through 6 hours of navigation in our simulated environment, we demonstrate that our proposed approach, unlike SGA, can reliably evaluate a large policy set. Each simulation 'epoch' consists of a random initialization of agent states followed by a 5 minute simulated run. 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 $\{\sigma_{p_x}, \sigma_{p_y}, \sigma_{|v|}, \sigma_\theta\} = \{10cm, 10cm, 10cm/s, 15°\}$. 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 and assume no angular velocity (ignoring any uncertainty). These uncertainties are propagated in the posterior state estimation $P(\boldsymbol{x}|\boldsymbol{z})$.

Our simulation experiments are run on an Intel i7 processor and 8GB RAM to mimic

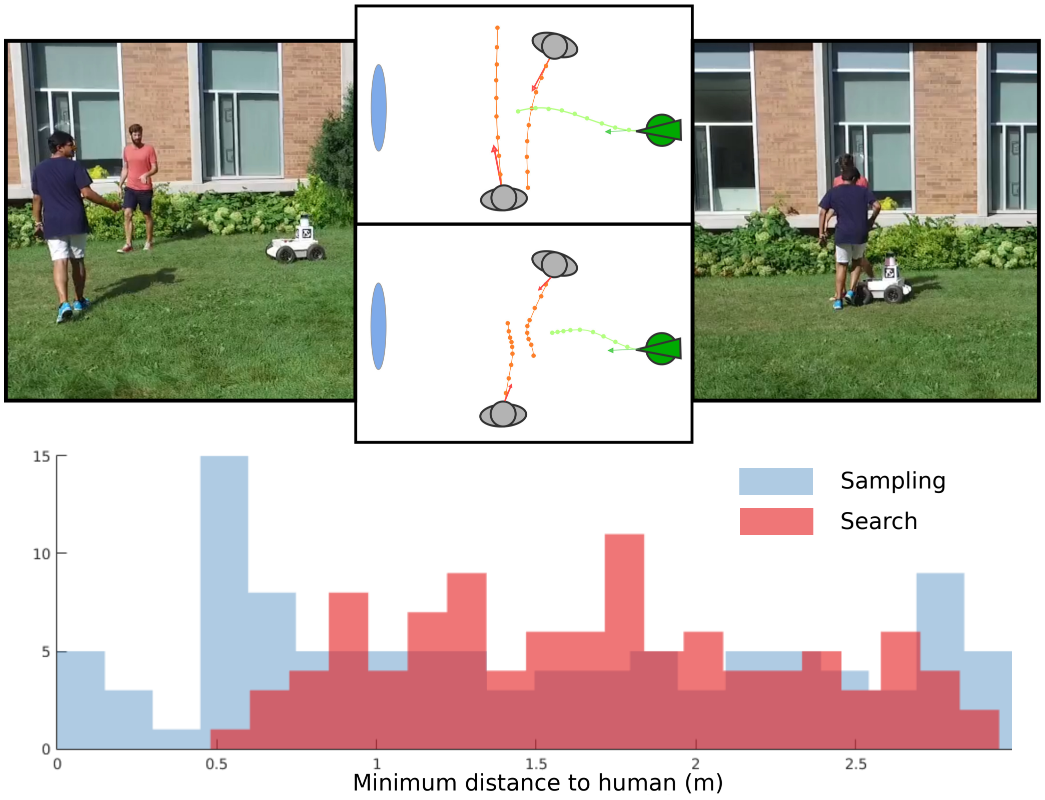


Figure 4.7: Random sampling often results in poor decision making. If the two humans move away from each other (*Middle-Top*), the robot would be able to pass through without causing inconvenience. However, if the agents move too close to each other and interact in the path of the robot (*Middle-Bottom*), *Go-Solo* is no longer a good policy for the robot. Initial configurations that result in such interactions should ideally be accounted for while evaluating the utility of a policy (Go-Solo, in this case). Such configurations have high probability density but low probability mass and hence, Monte-Carlo sampling from a cost-function agnostic posterior distribution often fails to capture these situations. *Left*: A real world scenario where two agents are walking towards each other. *Middle-Top*: At their current speed and orientation, the agents would pass the robot and not obstruct it. This is mostly the case. *Middle-Bottom*: However, the adverse situation where the two agents head straight at each other is also probable but is unlikely to be sampled (it has high probability density but low probability mass). *Right*: The scenario 0.7s later. The robot decides to stop at the very last moment when the probability mass for the imminent adverse scenario is large. *Bottom* The scenario described was repeated 20 times for each algorithm. The normalized histograms for the distance to the closest human the robot is moving towards. We notice that close calls and collisions are avoided by searching for likely high-cost outcomes. The sampling based approach collided with agent on 3 runs out of the 20 while our risk-aware approach never collided with another agent.

the computational capabilities of our robot. In order to react to sudden changes, MPDM relies on quick re-planning. The robot must replan every 300ms for effective real-time navigation. We evaluate the performance of risk-aware MPDM using 2 candidate sets of policies - a large candidate set with 10 policies, and a small set with 2 policies:

1. 2 Policies - $\{$*Go-Solo, Stop*$\}$ - The robot evaluates going straight towards the goal at maximum speed ($1.5m/s$) and stops if it senses danger.
2. 10 Policies - $\{$ *(Fast, Medium, Slow)*$\times$*(Straight, Left, Right), Stop*$\}$ - Rather than going straight towards the goal at maximum speed, the MPDM may also choose to go at *Medium* speed ($0.9m/s$) or *Slowly* ($0.2m/s$). Simultaneously, the robot can also choose to create a sub-goal to the *Left* or *Right* of the goal instead of going *Straight* to the goal as in *Go-Solo*.

We record the *Time Stopped* per goal reached, as well as the *Blame* normalized by the distance to the goal (*Blame per meter traveled*). *Time Stopped* indicates the failure of the planner to find a safe policy. With a larger policy set, the robot is more likely to find a safe policy, and *Stops* less often. However, if the robot cannot evaluate its policy set quickly enough, it is unable to react to sudden changes in the environment and accumulates *Blame*. Ideally, we would like a robot to navigate safely (low *Blame*), with minimal Stop-and-Go motion.

Fig. 4.6 shows how the inefficiencies in SGA become a performance bottleneck. While SGA can navigate safely (low *Blame*) with the small policy set, it often fails to find safe policies and stops. With 10 policies, SGA fails to find influential outcomes fast enough resulting in high *Blame*. Our proposed method, BP can reliably evaluate the large policy set in real-time, which significantly improves navigation performance.

When we slow down the simulator (three times slower than real-time) to allow an unrealistic planning time of $t_p = 1s$, SGA with 10 policies is able to drastically reduce *Blame*. However, our proposed approach, BP still outperforms SGA.

4.4.3 Real-World Experiments

We implemented our system on the MAGIC robot [144], a differential drive platform equipped with a Velodyne VLP-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. We use a laptop with an Intel i7 processor and 8GB RAM for our forward simulations and LCM [145] for inter-process communication.

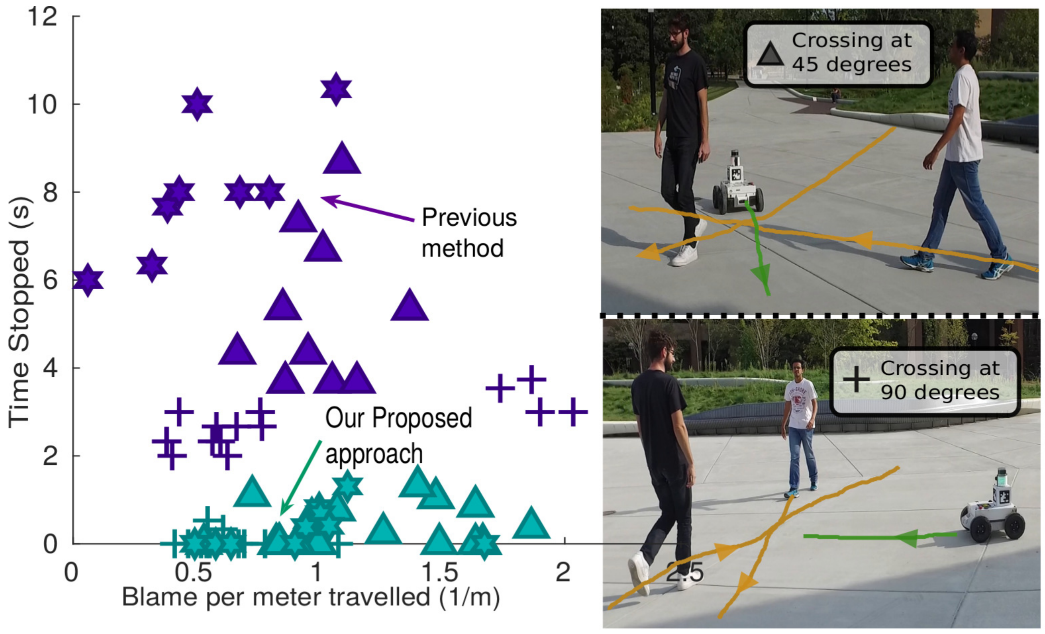


Figure 4.8: Repeated real-world experiments. We collect real-world data from three repeatable experiments represented by different symbols 1) pedestrians crossing orthogonally to the robot's trajectory (+), 2) pedestrians crossing the robot's path obliquely at 45 degrees (Δ) and 3) pedestrians walking slowly in front of the robot (star). We measure the *Time Stopped* for every goal reached as well as the *Blame per meter traveled* by the robot accumulated by inconveniencing pedestrians. Lower the *Time Stopped* and *Blame*, the better. Our proposed approach (green) can evaluate more policies in real-time than earlier possible. With more candidate policies, the robot can find good policies and can navigate safely without stopping unnecessarily.

Every 300ms (policy election cycle), MPDM chooses a policy. This constrains the robot to a budget of $N = 50$ forward simulations per core used (we use only one core for the planning algorithm, although the task is readily parallelizable, as discussed in Chapter 6). Although the policy election is slow, the robot is responsive as the policies themselves run at over 50Hz.

We ran two types of real-world experiments - one in which volunteers were asked to repeat structured scenarios designed to make multiple runs as similar as possible. and the second in which six volunteers were asked to move around spontaneously in the open space for 30 minutes (Fig. 4.1).

Fig. 4.7 shows the normalized histograms of the distance to the closest human, generated from 20 repetitions of a real-world scenario where two agents are walking towards each other and orthogonal to the robot's path. At their most-likely estimated speed and orientation, the agents would pass the robot and not obstruct it. However, in the adverse (and less likely) situation where the two agents head straight at each other, they would slow

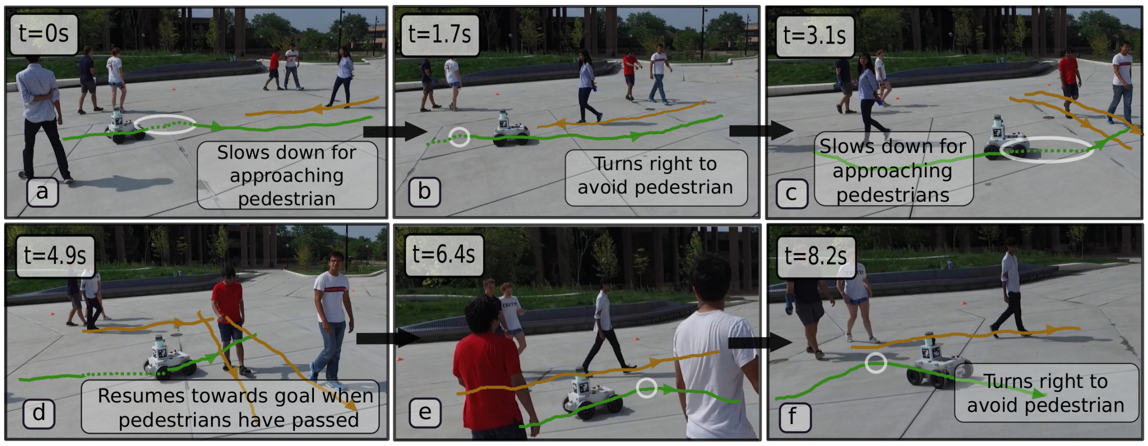

t=0s
Slows down for approaching pedestrian
a
t=1.7s
Turns right to avoid pedestrian
b
t=3.1s
Slows down for approaching pedestrians
c
t=4.9s
Resumes towards goal when pedestrians have passed
d
t=6.4s
e
t=8.2s
Turns right to avoid pedestrian
f

Figure 4.9: Real situations illustrating the benefits of risk-aware MPDM with many policies. The sequence of images depicts different situations that the robot encounters as it makes its way towards its goal. Dashed segments denote portions of the robot's trajectory (green lines) where it slowed down. The orange tracks represent the trajectories of relevant pedestrians. The lines were manually superimposed on the video after careful examination of the corresponding logs. The robot slows down upon discovering possible imminent collision (a and c) and turns appropriately to avoid them when possible (a and e) as denoted by the white ellipses. By dynamically switching between multiple policies, the robot is able to navigate safely, without stopping unnecessarily.

down, thereby obstructing the robot. MPDM with Monte-Carlo sampling (without searching for potentially dangerous outcomes) often results in poor decision making, resulting in close calls and collisions (peaks in the histogram). While the Monte-Carlo sampling failed to consistently capture influential configurations resulting in an optimistic utility for the *Go-Solo* policy, our risk-aware approach can consistently capture such configurations, making the robot stop when required.

Next, we present empirical results comparing two risk-aware approaches– Backprop-MPDM and SGA, on the three structured experiments described below (Fig. 4.8). Volunteers were asked to repeat the scenarios for a duration 15 minutes while the robot made its way towards its goal. The planning algorithm being executed was unknown to the participants.

1. Two pedestrians simultaneously cross the robot orthogonally. The pedestrians also approach each other as the robot approaches them, increasing the uncertainty and the scope for agent-agent interactions.

2. The pedestrians cross the robot's trajectory obliquely, beginning from the same side of the robot. This robot must anticipate how its interaction with the one pedestrian can affect its interaction with the other pedestrian. The outcome of this scenario is

sensitive to the pedestrian velocities.

3. Both pedestrians walk slowly in front of the robot.

As observed in simulation, SGA was too slow to evaluate the larger policy set reliably and was unsafe to deploy on our robot. Using SGA with two policies (purple), the robot fails to find safe policies and stops often. Our proposed method (green) can reliably evaluate 10 policies in real-time (similar Blame as compared to SGA with just two policies) and as a result, it is more likely to find safe policies (low *Time Stopped*).

In another experiment, seven volunteers were asked to move spontaneously between markers placed around the operating environment for 45 minutes. Fig. 4.9 demonstrates the emergent behavior from Backprop-MPDM through an 8 second time-line during this experiment[1]. With a rich set of candidate policies to choose from, the robot modulates its speed and heading to alleviate anticipated dangerous future outcomes.

4.5 Summary

In this chapter, we introduced the idea of discovering influential outcomes for MPDM, skew decision-making to award policies that have fewer potentially dangerous outcomes. A key advantage is that the evaluation of a policy can be seen as an optimization problem, as opposed to the computation of a probabilistic expectation.

By recognizing that the kinematic models used for the agents have an impact on the quality of the gradients, we showed how accurate gradients of a forward simulation can be computed efficiently using backpropagation. Using these gradients, we can quickly evaluate a large number of discrete policies *reliably* in real-time resulting in significant performance improvements.

[1]We encourage the reader to see our video (https://www.youtube.com/playlist?list=PLbPJN-se3-QiwIITl5cNsUV4-SRIyl9OM) demonstrating the advantages of risk-aware MPDM.

CHAPTER 5

C-MPDM: Continuously-parameterized risk-aware MPDM by quickly discovering contextual policies

5.1 Introduction

There is a core tension in MPDM type systems — it is desirable to add more policies to increase the expressivity of the system for better emergent behavior, however, this increases computational cost. Risk-aware MPDM evaluates each candidate policy (*ego-policy*) by anticipating potentially dangerous outcomes through an on-line optimization process based on forward roll-outs. More specifically, each ego-policy π_r is evaluated based on the most influential (likely high-cost) forward simulated outcome $\Psi(\pi_r)$ that may occur which is discovered by optimizing a probabilistic cost surface -

$$\Psi(\pi_r) = \arg\max_{\boldsymbol{x}_0 \in \mathrm{X}} \{P(\boldsymbol{x}_0) C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)\}. \tag{5.1}$$

This optimization is computationally expensive and only a handful of policies can be evaluated reliably in real-time.

Policy election for risk-aware MPDM can be modeled as a real-time bilevel program, where during each planning cycle (every 300ms), the ego-policy with the most benign influential outcome is chosen and executed. Bilevel optimization is a well-studied class of mathematical programs encountered in various fields ranging from management [42], to optimal control [43] where there are two levels of optimization tasks, one nested within the other. In risk-aware MPDM, the upper-level optimizer (the ego-robot) chooses the policy with the most benign (low-cost) evaluation, while lower-level optimization (risk-aware policy evaluation of an ego-policy) involves finding the most potentially dangerous (likely, high-cost) outcome from all possible pedestrian configurations. In this way, risk-

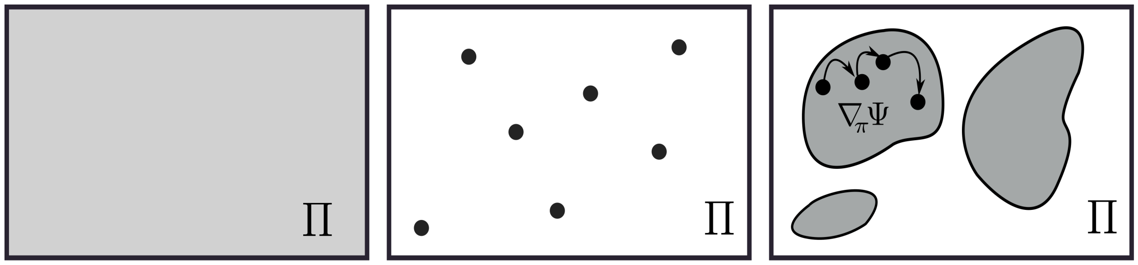


Figure 5.1: Expressible policy spaces. *Left*: In principle, we would like to find the best policy from the entire space of policies Π; this is generally intractable. *Middle*: Earlier MPDM systems constrained the search to a finite number apriori-known discrete policies, whose size (perhaps 5-10) depends on the computational budget. *Right*: Continually-parameterized MPDM (C-MPDM) can represent much larger volumes within the policy space. By quickly generating promising context-derived candidate policies using "risk-aware policy-gradients" $\nabla_\pi \Psi$, C-MPDM increases expressivity of the actions available to the robot without increasing computational complexity.

aware MPDM can be viewed as a bilevel optimization -

$$\begin{aligned} &\min_{\pi_r \in \Pi} P(\boldsymbol{x}_0) C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big) \\ &\text{s.t. } \boldsymbol{x}_0 = \arg\max_{\boldsymbol{x}_0 \in \mathrm{X}} \{P(\boldsymbol{x}_0) C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)\} \\ \iff &\boldsymbol{x}_0 = \Psi(\pi_r). \end{aligned} \tag{5.2}$$

In our bilevel program (Eqn. 5.2), even the lower level optimization (computing $\Psi(\pi_r)$ exactly for fixed ego-policy) is computationally infeasible due to the large space of possible initial pedestrian configurations [149]. As a result, earlier MPDM systems used a discrete set of hand-crafted policies for the upper-level optimization to reduce the bilevel program to a set of simpler optimizations.

Algorithm 3 describes earlier approaches to policy-election. Provided with a probability distribution over initial configurations, $P(\boldsymbol{x}_0)$, an apriori-known discrete set of candidate policies, Π_d, and a forward simulation budget, N_π, each candidate policy evaluated (scored) independently (Line 2) according to the most influential (worst-case) outcome discovered within the computational budget and the policy with the most benign influential outcome is elected. Only a handful of ego-policies can be reliably evaluated in real-time, limiting Alg. 3 to a small number of discrete policies — a significant performance bottleneck. Moreover, the system designer must decide the candidate ego-policy set apriori, and only a small subset may be relevant at any time.

In the next section, we extend the risk-aware MPDM framework by allowing policies to have continuous-valued parameters. We propose Continuously-parameterized Multi-Policy

Algorithm 3 Policy Election with Handcrafted Policies

```
1: function POLICY-ELECTION LOOP($P(\boldsymbol{x}), \Pi_d, N_\pi$)
2:     for $\boxed{\pi \in \Pi_d}$ do
3:         Initialize $U_\pi, n \leftarrow 0$
4:         while $n < \frac{N_\pi}{|\Pi_d|}$ do
5:             Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x})$
6:             $U^*, n_{opt} \leftarrow$ Optimize-Env($\boldsymbol{x}_0, \pi$)
7:             $n \leftarrow n + n_{opt}$
8:             $U_\pi \leftarrow \max\{U^*, U_\pi\}$
9:         end while
10:    end for
11:    $\pi^* \leftarrow \arg\min_\pi U_\pi$
12: end function
```

Decision Making (C-MPDM) to radically enhance the expressivity of MPDM without increasing computational complexity. By allowing policies to have *continuous*-valued parameters, and then efficiently computing good values of those continuous parameters (Fig. 5.1), C-MPDM enables the robot to choose from an infinite number of policies in real-time.

To achieve this, C-MPDM has to overcome many practical challenges. Merely checking strict or local optimality in bilevel optimization problems is NP-hard and most academic research has been focused on simple bilevel programs with convex objective functions [150]. In our domain, even evaluating the cost function involves a time-consuming forward simulation. Furthermore, the high-dimensional search-space of possible robot policies and pedestrian configurations and the complex multi-agent interactions make the forward-simulated trajectories and thereby the cost function sensitive to the decision variables. The conflicting objectives, the real-time requirements and the lack of a closed-form expression for the objective function makes this problem harder.

5.2 Contributions

Our proposed algorithmic approach fundamentally re-thinks the MPDM algorithm, replacing a core component with a method that more closely resembles a deep learning algorithm than a motion planner (though it is neither!). The key idea is to quickly generate promising context-derived candidate policies using a local iterative gradient-based search procedure (Fig. 5.1) where the necessary gradients are efficiently computed using backpropagation. Promising policy candidates are then extensively evaluated via the forward roll-out process described in the previous chapter. Our contributions include the following:

1) We formulate C-MPDM as a bilevel optimization and allow policies to have continuous-

valued parameters, which improves the expressivity and flexibility of the decision making process.

2) We provide an effective real-time solution to the bilevel program. Our novel anytime algorithm finds increasingly desirable contextual ego-policy parameters.

3) Through extensive experiments in simulation and on a real robot platform, we demonstrate the benefits of C-MPDM over other approaches such as evaluating a fixed set of hand-crafted policies and random policy sampling.

4) We compare the performance of C-MPDM with a state-of-the-art trajectory based planner, Model-Predictive Equilibrium-Point Control (MPEPC) [44] through extensive real-world experiments. Using objective metrics and subjective feedback, we demonstrate that C-MPDM produces emergent behavior that is more reliable than MPEPC for autonomous navigation in dynamic environments with large uncertainty.

5.3 Method

Previously, MPDM used a discrete set of domain-specific hand-crafted policies to navigate social environments. For example, in the indoor hallway setting, we used $\Pi = \{\textit{Go-Solo}, \textit{Follow}_j, \textit{Stop}\}$. We allow the robot's policy π_r (which we refer to as *ego-policy*) to be an instantiation of a continuously-parameterized policy $\pi(v_{pref}, \psi_{g_r}) \in \Pi$.

The robot executing a policy $\pi_r = \pi(v_{pref}, \psi_{g_r})$ tries to move at a preferred speed v_{pref}, while avoiding obstacles according to the Social Force Model [23]. Rather than heading straight towards its goal g_r, the robot tries to move towards a point to the left or right of the goal, as determined by the direction offset parameter ψ_{g_r}. For example, $\pi_r = \pi(0.5, 30°)$ implies a policy where the robot tries to move at 0.5m/s at an orientation $30°$ to the right of the goal. For the sake of clarity, we do not explicitly refer to parameters of an ego-policy π_r. Gradients are expressed w. r. t. π_r, although we are implicitly referring to its parameters.

We propose a novel approach where accurate gradients are computed efficiently through backpropagation and these gradients are used quickly discovering effective contextual ego-policy parameters (boxed in Alg. 4). Each contextual candidate ego-policies is extensively evaluated. Our anytime candidate generation algorithm produces increasingly desirable policies for the robot to execute.

Generating Contextual Risk-Aware Policies

Deep neural networks chain (compose) relatively simple functions such as convolutions or sigmoids to model complex functions. In the same way, a forward simulation for H

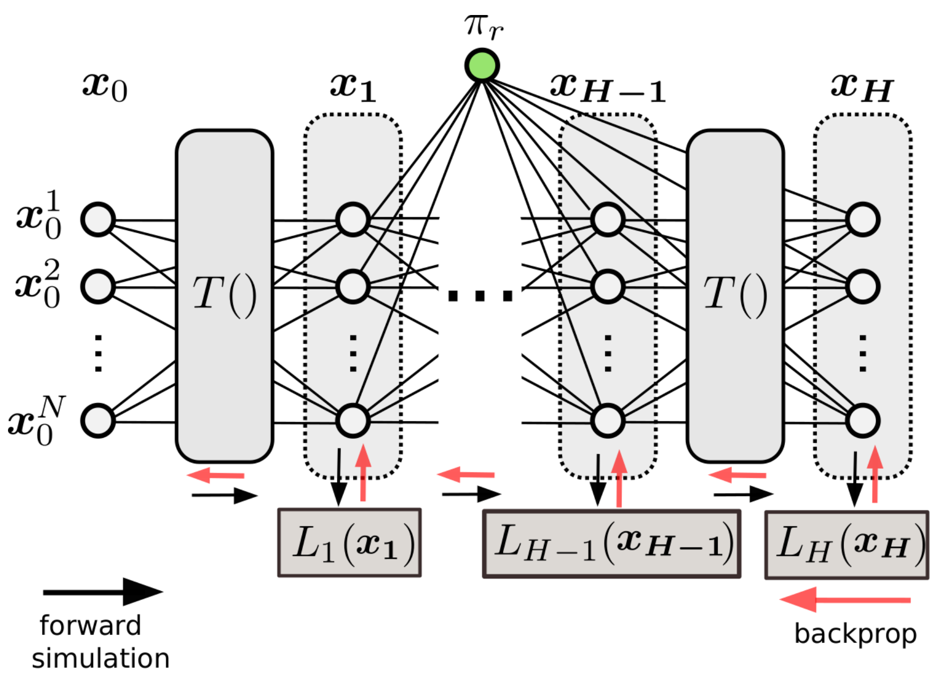


Figure 5.2: A deep network representation for our cost function. The initial configuration $\boldsymbol{x}_0$ propagates through several layers, each representing the transition function T. The output of layer t determines a cost for a single time-step $L_t(\boldsymbol{x}_t)$. Our cost function $C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))$ accumulates costs calculated at each time-step along the forward simulated trajectory.

time-steps captures the complex dynamics of the system using simple one-step transition functions T. High-cost forward-simulated outcomes correspond to those where the robot inconveniences other agents by driving too close to them, thus accumulating high *Blame*. We want to penalize trajectories that have bad interactions at any point during the forward roll-out, not just those that *end* in a bad interaction. The robot is also rewarded according to the *Progress* it makes towards the goal. Therefore, unlike most deep networks, the cost function $C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))$ is not a simple loss computed at the final output, but rather costs accumulated at each time step $L_t(\boldsymbol{x}_t)$.

We define a partial cost function $\mathrm{L}(t, \boldsymbol{X})$ that accumulates costs along the trajectory from time $\tau = t \ldots H$ (until the end of the roll-out).

$$\mathrm{L}(t, \boldsymbol{X}) = \sum_{\tau=t}^{H} L_\tau(\boldsymbol{x}_\tau). \tag{5.3}$$

Accurate gradients of the cost function can be computed efficiently by backpropagation through deep network, which leverages the following recurrence relation -

$$\begin{aligned}\nabla_{\boldsymbol{x}_t}\mathrm{L}(t,\boldsymbol{X}) =& \frac{\partial \mathrm{L}(t,\boldsymbol{X})}{\partial \boldsymbol{x}_t} = \frac{\partial\{\mathrm{L}(t+1,\boldsymbol{X}) + L_t(\boldsymbol{x}_t)\}}{\partial \boldsymbol{x}_t} \\ =& \frac{\partial \mathrm{L}(t+1,\boldsymbol{X})}{\partial \boldsymbol{x}_{t+1}}\frac{\partial T(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} + \frac{\partial L_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}. \end{aligned} \tag{5.4}$$

The gradient of the cost function with respect to the agent configurations $\nabla_{\boldsymbol{x}_0} C(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r)) = \nabla_{\boldsymbol{x}_0}\mathrm{L}(0,\boldsymbol{X})$ is used during gradient-ascent to simultaneously perturb the configurations of all pedestrians towards increasingly likely and high-cost outcomes until convergence as follows -

$$\boldsymbol{x}_0 = \boldsymbol{x}_0 + \eta_1\Big(\frac{\nabla_{\boldsymbol{x}_0} C(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r))}{C(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r))} + \frac{\nabla_{\boldsymbol{x}_0} P(\boldsymbol{x}_0)}{P(\boldsymbol{x}_0)}\Big). \tag{5.5}$$

Each agent's update rate is determined using line-search along the gradient direction. Note that Eqn. 5.5 locally optimizes $\log\big(P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi_r,\boldsymbol{x}_0)\big)\big)$. We refer to this procedure as Optimize-Env.

The key insight is that in addition to discovering influential outcomes, we can use the same backpropagation machinery to compute the gradient of the cost function with respect to the ego-policy parameters -

$$\nabla_{\pi_r} C(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r)) = \sum_{t=1}^{H}\frac{\partial \mathrm{L}(t,\boldsymbol{X})}{\partial \boldsymbol{x}_t}\frac{\partial \boldsymbol{x}_t}{\partial \pi_r}.$$

We perform gradient-descent, perturbing the ego-policy π_r towards increasingly benign parameters until convergence as follows -

$$\pi_r = \pi_r - \eta_2\frac{\nabla_{\pi_r} C(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r))}{C(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r))}. \tag{5.6}$$

This gradient-descent procedure, which we call Optimize-Robot, locally optimizes $\log\big(C\big(\boldsymbol{X}(\boldsymbol{x}_0,\pi_r)\big)\big)$.

Since finding the global optimum for the bilevel program (Eqn. 5.2) is computationally intractable, we first generate promising contextual candidate policies using a greedy local search procedure (POLICY-GENERATION) and then evaluate these candidates extensively (POLICY-EVALUATION).

Algorithm 4 summarizes the overall policy election cycle. C-MPDM elects an ego-policy, taking as input a probability distribution over initial pedestrian configurations, $P(\boldsymbol{x}_0)$, a continuous ego-policy space, Π, and forward simulation budgets for candidate generation (N_{cg}) and policy evaluation (N_π). A pedestrian configuration $\boldsymbol{x}_0$ sampled from $P(\boldsymbol{x}_0)$ and

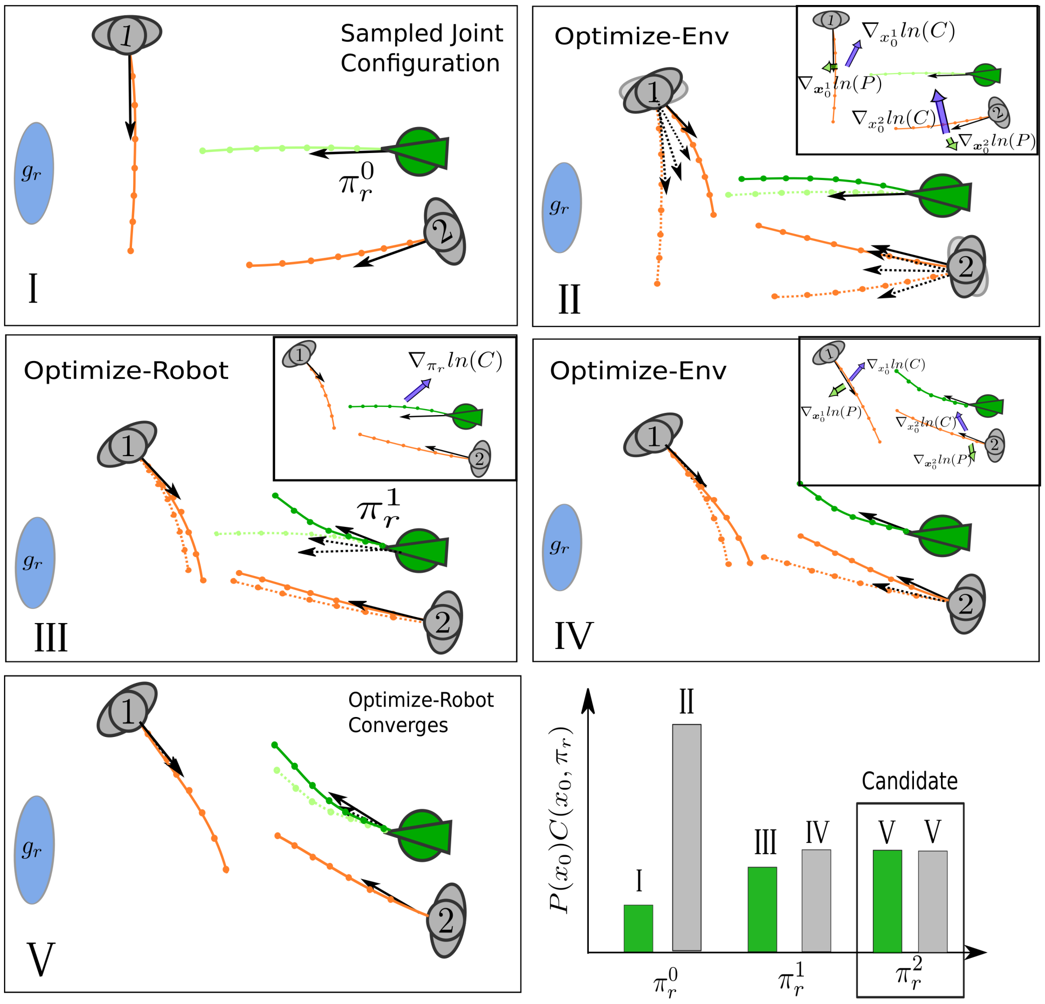


Figure 5.3: Candidate ego-policy generation through iterative gradient-based optimization (Alg. 4 - POLICY-GENERATION). The forward propagated outcome of a randomly sampled joint configuration (tile I) is not discouraging for the robot as its ego-policy π_r^0 does not inconvenience either agent. For agent $i \in \{1, 2\}$, the gradients computed using backpropagation $\nabla_{\boldsymbol{x}_0^i} ln\big(C(\boldsymbol{X})\big)$ (Blue) drives the agent towards a configuration where the robot would cause inconvenience while $\nabla_{\boldsymbol{x}_0^i} ln\big(P(\boldsymbol{x}_0)\big)$ (Green) drives it towards a more likely configuration (tile II inset). The agent configurations are simultaneously updated (dotted black arrows) according to Eqn. 5.5 until convergence, resulting in a likely high-cost outcome (tile II) that is used to evaluate π_r^0. The bar plot (*Bottom-Right*) shows the corresponding rise in the objective function $P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi_r^0, \boldsymbol{x}_0)\big)$. Once π_r^0 is evaluated, $-\nabla_{\pi_r} ln\big(C(\boldsymbol{X})\big)$ is computed (tile III inset) through backpropagation (Blue), and the ego-policy is updated according to Eqn. 5.6 until convergence, resulting in a more benign robot policy π_r^1 with a lower objective function value. The newly discovered ego-policy π_r^1 is then evaluated once again perturbing the agent configurations to discover the locally most influential outcome for π_r^1 (tile IV). The bar plot (*Bottom-Right*) shows that even though the outcome in tile III for π_r^1 has a higher cost than the outcome (tile I) for π_r^0, the worst-case outcome for π_r^1 (tile IV) is more benign than that for π_r^0 (tile II). After one more iteration, our approach converges to a saddle point and a promising candidate policy π_r^2.

an ego-policy π_r sampled from Π (Line 3) seed the POLICY-GENERATION procedure (Line 4) which discovers promising policy parameters in the current navigation context. Promising policies π_r^* are extensively evaluated (Line 8) and the policy with the most benign influential outcome is executed (Line 10).

POLICY-GENERATION is an iterative gradient-based process that converges at a saddle point where the robot's policy and the pedestrian configuration are both locally optimal as illustrated in Fig. 5.3. First, Optimize-Env (Line 15) perturbs the seed pedestrian configuration $\boldsymbol{x}_0$ towards the locally most influential outcome which determines the score U_0 of the seed ego-policy π_r. Then, the seed ego-policy is repeatedly perturbed and evaluated locally - at each iteration, Optimize-Robot (Line 19) perturbs the ego-policy π_r towards more benign parameters using n_{opt} gradient-descent steps (Eqn. 5.6) and the perturbed policy is evaluated according to the locally most influential outcome (Line 21). If the forward simulation budget for candidate generation N_{cg} is exceeded (Line 18) or if ego-policy perturbation does not improve the worst-case outcome by ϵ (Line 24), the search terminates. If perturbations in the ego-policy result in a significantly more benign (locally) worst-case outcome than the seed ego-policy (Line 5), we assume that π_r^* is likely to be promising.

The POLICY-EVALUATION function evaluates π_r^* more extensively by performing local searches for influential outcomes from different seed pedestrian configurations (Line 36) using at most N_π forward simulations. Otherwise, a new joint configuration is sampled and the procedure repeats.

Connection with Policy Gradient Methods

In reinforcement learning (RL), an agent (the robot) learns to act in an environment so as to maximize its cumulative reward. The robot's policy induces a distribution of trajectories in the environment and the goal of an RL algorihtm is to find a policy for the robot with the best expected return under this distribution. The main challenge faced by RL algorithms is to converge to a good policy quickly (using few samples). Monte-Carlo sampling based on the on-policy trajectory distribution can result in high-variance estimates of the expected return. This has motivated the use of importance sampling based on carefully constructed behavior policies for sample-efficient policy evaluation [151]. State-of-the-art off-policy actor-critic algorithms for reinforcement learning such as TRPO [152] and A3C [153] cycle between policy evaluation and policy improvement. These methods are more sample-efficient because they re-use trajectories generated from behavior policies for evaluating the target policy by assigning importance weights to each trajectory based on its likelihood ratio and relative return.

Algorithm 4 Policy Election for Continuous Policy Spaces

```
 1: function C-MPDM($P(\boldsymbol{x}), \Pi, N_{cg}, N_\pi$)
 2:     while Time Remaining do
 3:         Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x}),\ \ \pi_r \sim \Pi$
 4:         $\pi_r^*, U_\pi, U_0 \leftarrow$ Policy-Generation$(\pi_r, \boldsymbol{x}_0, N_{cg})$
 5:         if $U_\pi - U_0 \leq \delta$ then
 6:             continue
 7:         end if
 8:         $U_\pi \leftarrow$ Policy-Evaluation$(\pi_r, U_\pi, N_\pi)$
 9:     end while
10:     return $\arg\min_\pi U_\pi$
11: end function
12:
13: function $\boxed{\text{POLICY-GENERATION}}$ $(\pi_r, U_\pi, N_{cg})$
14:     Initialize $U_\pi, n \leftarrow 0$
15:     $\boldsymbol{x}_0, n_{opt} \leftarrow$ Optimize-Env$(\boldsymbol{x}_0, \pi_r)$
16:     $n \leftarrow n + n_{opt}$
17:     $U_0 \leftarrow P(\boldsymbol{x}_0)C(\boldsymbol{x}_0, \pi_r)$
18:     while $n < N_{cg}$ do
19:         $\tilde{\pi}_r, n_{opt} \leftarrow$ Optimize-Robot$(\boldsymbol{x}_0, \pi_r)$
20:         $n \leftarrow n + n_{opt}$
21:         $\boldsymbol{x}_0, n_{opt} \leftarrow$ Optimize-Env$(\boldsymbol{x}_0, \tilde{\pi}_r)$
22:         $n \leftarrow n + n_{opt}$
23:         $U_i \leftarrow P(\boldsymbol{x}_0)C(\boldsymbol{x}_0, \tilde{\pi}_r)$
24:         if $U_i \leq U_{i-1} - \epsilon$ then
25:             $\pi_r \leftarrow \tilde{\pi}_r$
26:             $U_\pi \leftarrow$ U$_i$
27:         else
28:             break
29:         end if
30:     end while
31:     return $\pi_r, U_\pi, U_0$
32: end function
33:
34: function POLICY-EVALUATION$(\pi_r, U_\pi, N_\pi)$
35:     while $n < N_\pi$ do
36:         Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x})$
37:         $U^*, n_{opt} \leftarrow$ Optimize-Env$(\boldsymbol{x}_0, \pi_r)$
38:         $n \leftarrow n + n_{opt}$
39:         $U_\pi \leftarrow \max\{U^*, U_\pi\}$
40:     end while
41:     return $U_\pi$
42: end function
```

MPDM is a coupled behavioral planning and intent estimation framework where *all* agents (pedestrians and the robots) are executing policies and the joint policies induce a distribution over trajectories. MPDM simulataneously reasons over the policies of multiple agents, but each agent's policy is more structured and has fewer parameters (uses more domain knowledge) than policies typically learned by RL algorithms. Unlike traditional RL settings, in our domain the complexity arises from the closed-loop interactions between agents in the environment, which makes the trajectories and cost-function sensitive to the agents' behaviors (environmental state). In section 4.1, we showed that approximating the expected utility for a particular ego-policy with MPDM's real-time constriants is unreliable and therefore, we evaluate policies by quickly discovering the worst-case pedestrian configuration.

Even though the notions of policy-evaluation and policy-improvement in C-MPDM are similar to those in actor-critic methods for RL, our ego-policy's 'critic' is the joint pedestrian configuraiton with the worst-case outcome, which is local and highly adaptive with respect to the robot's policy parameters. Moreover, the importance weights for re-using off-policy trajectories would have high variance our ego-policies are deterministic and the trajectories are very sensitive to the policy. Thus, C-MPDM tackles new practical challenges arising due to our spiky cost function and real-time constraints and we take a different optimization approach more closely resembling a Stackelberg game between the pedestrians and the robot.

Connection with Optimal Control

Optimal control attempts to find the best sequence of actions for a system with known dynamics and limited stochasticity within a high-dimensional action space. Classic techniques for trajectory optimization and optimal control such as LQR for linear systems [154], or nonlinear approaches such as iLQG [155] and LQR-Trees [156], use backpropagation to efficiently update solutions. While our proposed method uses a similar backpropagation process, our application of these gradients is different. C-MPDM tries to find optimal policy for the robot given worst-case policies of other agents, making it a bilevel optimization problem.

5.4 Results

Our operating environment consists of an open area that is freely traversed by a set of pedestrians that randomly change speed and direction without signaling while the robot tries to reach its goal. The unconstrained nature of this domain makes the trajectories more sensitive to the initial pedestrian configurations as well as the robot's ego-policy. By re-planning quickly (every 300ms), the robot is able to react to sudden and unexpected changes in the environment.

For each observed agent i, the robot maintains a probabilistic estimate of its state. A sudden change in speed is accounted for by assuming a distribution over the preferred speed v_{pref} of each agent that is a mixture of two truncated Gaussians - one centered around the estimated most-likely current speed with a $\sigma = 0.4m/s$ to account for speeding up or slowing down and a truncated half Gaussian with a peak at 0 and $\sigma = 0.2m/s$ to account for coming to a sudden stop. Pedestrian i's sub-goal g_i is inferred from a set of salient points in the environment using a Naive Bayes classifier. However, the pedestrian can suddenly change direction without signaling which is captured by assuming a Gaussian distribution for the pedestrian's short-term heading ψ_{g_i} that is centered around the agent's estimated most-likely orientation and $\sigma = 30°$. All truncated Gaussians are restricted to $\mu \pm 1.5\sigma$.

5.4.1 Simulation Experiments

Our simulation experiments are run on an Intel i7 processor and 8GB RAM to mimic the computational capabilities of our robot platform. Our simulation environment is an open space, freely traversed by a set of agents while the robot tries to reach a goal. Agents can randomly slow down or come to a stop without signaling. First, we demonstrate that our proposed approach discovers more desirable ego-policies than random policy sampling. Next, we show that C-MPDM outperforms MPDM with a discrete set of apriori hand-crafted policies by finding safer, and more effective policies in real-time.

Efficiency of Candidate Generation

We generated a dataset consisting of 4k randomly chosen simulated scenarios, each of which has at least one agent present within 5m of the robot. For each scenario, we estimate $\max\limits_{\pi_r \in \Pi} \Psi(\pi_r)$, the optimal solution to the bilevel optimization (Eqn. 5.2) by evaluating a large number of policies $\tilde{\Pi}$, consisting of 1k randomly sampled ego-policies as well as 500 context-aware policies generated using our proposed approach (Alg. 4). For each policy, $\Psi(\pi_r)$ was estimated using the POLICY-EVALUATION function with a forward simulation budget $N_\pi = 100$. This brute force estimation of the "globally optimal" ego-policy takes about 5 minutes for each scenario (three orders of magnitude slower than our real-time requirements).

Given real-time constraints, only a handful of ego-policies can be evaluated reliably.

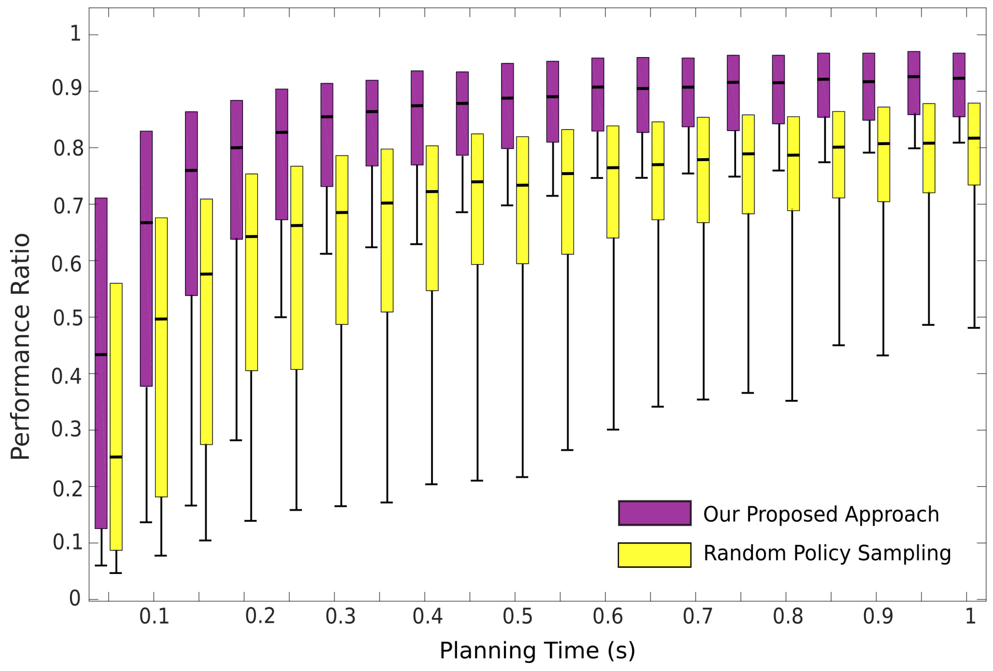


Figure 5.4: Varying the planning time, we evaluate the efficacy of our proposed method in estimating $\min\limits_{\pi_r \in \Pi} \Psi(\pi_r)$ (a bilevel optimization). We compare the distribution of the *Performance Ratio* of 400k candidate sets generated using our approach with those generated by random policy sampling. The bars mark the median and quartiles of the data-points while the bar represents the 10^{th} percentile. A *Performance Ratio* of 1 indicates optimality, while candidates with lower *Performance Ratios* would cause the robot to stop unnecessarily or pick a sub-optimal policy. Random policy sampling often fails to find desirable ego-policy parameters even with impractical planning times of $1s$. Our method significantly outperforms random policy sampling over the entire range of *Planning Time*.

For a particular scenario in the dataset, we define the *Performance Ratio* of N candidate ego-policies $\Pi_{cand} = \{\pi_r^i\}_{i=1}^N$ the ratio of "globally optimal" ego-policy's utility to the utility of the most benign ego-policy in the generated candidate set -

$$\text{Performance Ratio}\,(\Pi_{cand}) = \frac{\min\limits_{\pi_r \in \tilde{\Pi}} \Psi(\pi_r)}{\min\limits_{\pi_r \in \Pi_{cand}} \Psi(\pi_r^i)}.$$

A *Performance Ratio* of 1 is ideal and smaller ratios imply poorer candidates.

Varying the time available for planning t_p from $50ms$ to $1s$, we compare the *Performance Ratio* of 400k candidate sets of randomly sampled ego-policies with context-derived candidate ego-policies generated using our proposed approach. As more planning time is available, more candidate policies can be evaluated and in general, the *Performance Ratio* increases. However, in order to stay reactive to sudden changes in the environment, t_p

should not exceed $400ms$ (based on experiments on our physical robot platform).

For each scenario, candidates are bootstrap sampled from the dataset and their *Performance Ratio* is represented by box-plots in Fig. 5.4. The boxes represent the quartiles while the bar represents the 10^{th} percentile. We observe that our method significantly outperforms random policy sampling over the entire range of *Planning Time*. Within real-time constraints ($300 - 400ms$), while random policy sampling is highly unreliable (long 10^{th} percentile bars), our iterative gradient-based approach the elected policy is almost always within a factor of two of the optimal ego-policy (99.9% of the times) which is better than random policy sampling with $1s$ of (impractical) planning time.

C-MPDM System Validation

Through 10 hours of autonomous navigation in our simulated environment, we demonstrate that the enhanced expressivity provided by C-MPDM results in significant performance improvements. Each simulation 'epoch' consists of a random initialization of agent states followed by a 5 minute simulated run at a granularity $\Delta t = 0.15$s. During each policy-election cycle, the simulator was perturbed to account for sensor noise and tracking uncertainty.

We record the *Time Stopped* as well as the *Blame* normalized by the distance to the goal. *Time Stopped* indicates the failure of the planner to find a safe policy. With a larger policy set, the robot is more likely to find a safe policy, and *Stops* less often. However, if the robot cannot evaluate its policy set quickly enough, it is unable to react to sudden changes in the environment and accumulates *Blame*. Ideally, we would like a robot to navigate safely (low *Blame*), with minimal Stop-and-Go motion.

We run the simulator both in real-time ($t_p = 0.3s$), as well as slowed-down to allow an unrealistic planning time ($t_p = 1.5s$). We compare the performance of our proposed approach C-MPDM, which allows for continuous-valued parameterized policies with the following alternatives for generating discrete policies -

1. *Hand-crafted Candidates (HC)* - A set of hand-crafted candidate policies $\Pi_{hc} = \{$*(Fast, Medium, Slow)*$\times$*(Straight, Left, Right), Stop, Follow-other*$\}$ - used in previous MPDM-systems [149] is provided. Each candidate is independently evaluated as in Alg. 3. Rather than going straight towards the goal at maximum speed ($1.6m/s$), the robot may also choose to go at *Medium* speed ($0.9m/s$) or *Slowly* ($0.2m/s$). Simultaneously, the robot can also choose to create a sub-goal to the *Left* or *Right* of the goal. Additionally, the robot can choose to follow any nearby agent. Due to the open nature of our domain, the *Follow-other* policies have the lowest priority (they

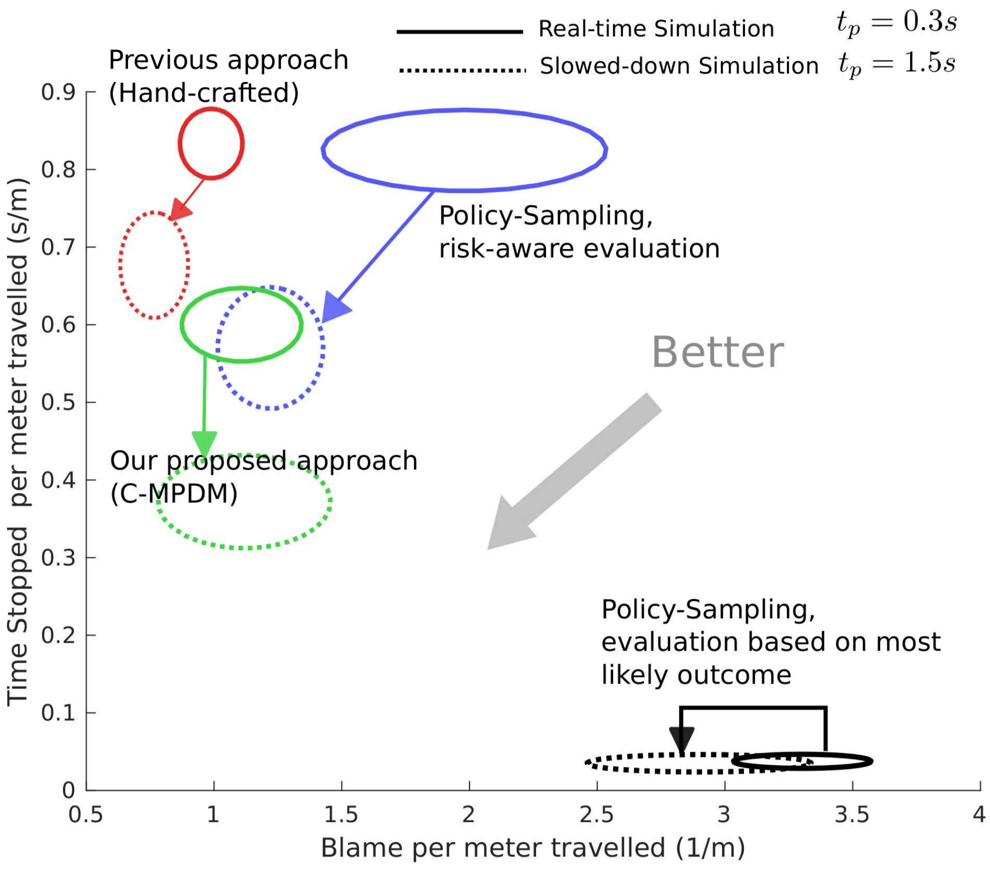


Figure 5.5: Our proposed method, C-MPDM can find good policies more often, even other where methods cannot. We compare the performance of various algorithms on 10 hours of navigation in our simulated environment. We measure the *Time Stopped* for every goal reached as well as the *Blame per meter traveled* by the robot. For each algorithm, we use bootstrap sampling to estimate the mean and standard error for these metrics, represented by the axes of an ellipse. Smaller values of *Blame* and *Time Stopped* are better. Note that in practice, a planning time $t_p < 0.4s$ is required for responsive behavior $t_p = 1.5s$ is impractical for our application. Without risk-aware evaluation, even densely sampling the policy space fails to anticipate potentially dangerous outcomes and incurs high *Blame*. Upon running the simulator in real-time with a planning time $t_p = 0.3s$ (solid ellipses), we observe that our proposed method (C-MPDM) outperforms both, risk-aware policy evaluation with random samples and with hand-crafted ego-policy parameters. Upon slowing down the simulator in order to allow an impractical planning time $t_p = 1.5s$ (dashed ellipses), all the algorithms perform better. Still, C-MPDM outperforms both policy-sampling and Hand-crafted policies.

are evaluated only if time permits after evaluating the others).

2. *Policy Sampling with Risk-Aware evaluation (PS-RA)* - Given a continuous policy space Π, (without a set of hand-crafted policies), a set of policies can be randomly sampled from Π and evaluated in a risk-aware fashion.

3. *Policy Sampling without Risk-Aware evaluation (PS-ML)* - Since risk-aware policy evaluation is computationally expensive, an alternate approach is to densely sample the policy space Π and evaluate policies very quickly only based on the most likely scenario. In our experiments, around 150-200 policies could be evaluated for a planning time of $0.3s$.

For each of the four planning algorithms, Fig. 5.5 shows the mean and standard error for the *Time Stopped* and the *Blame per meter traveled* by the robot. Under real-time constraints, we observe that our proposed method (C-MPDM) is able to discover effective ego-policy parameters more often than both, risk-aware policy evaluation with randomly sampled (PS-RA) as well as with hand-crafted (HC) ego-policy parameters and *Stops* less frequently. On the other hand, dense policy sampling (PS-ML) does not account for the uncertainty while evaluating a policy and accumulates much higher *Blame* as it fails to anticipate potentially dangerous outcomes and *Stops* only when collision is imminent. Upon slowing down the simulator to 5x slower than real-time constraints would allow, sampling performs much better. The hand-crafted *Follow-other* policies can be evaluated with $t_p = 1.5s$ and hence, with more candidates, HC *Stops* less often. However, C-MPDM still offers much more flexibility to the decision making process and outperforms both HC and PS-RA.

5.4.2 Real-World Experiments

We implemented our system on a skid-steer robot equipped with a Velodyne VLP-16 laser scanner used for tracking pedestrians as well as for localization. Every 300ms, MPDM evaluates a set of policies and chooses the least risky one. Although the policy election is slow, the robot is responsive as the policies themselves run at 50Hz. Fig. 5.6 shows data from 75 minutes of repeatable real-world experiments where volunteers were asked to repeat two fixed scenarios while the robot made its way towards its goal about $25m$ away -

1. Two pedestrians simultaneously cross the robot orthogonally. The pedestrians also approach each other as the robot approaches them, increasing the uncertainty and the scope for agent-agent interactions.

2. Both pedestrians walk in front of the robot, flanking it from either side.

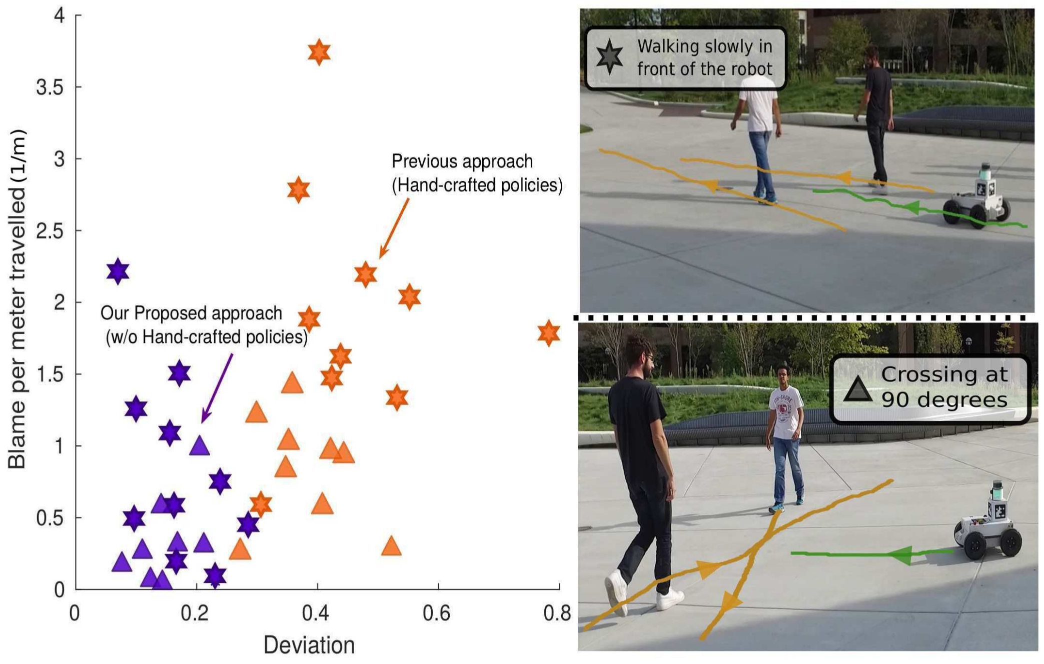


Figure 5.6: Repeated real-world experiments. Data is collected from two repeatable experiments represented by different symbols. 1) Pedestrians crossing the robot's trajectory orthogonally (Δ) and 2) Pedestrians walking slowly in front of the robot (star). We measure the trajectory *Deviation* as well as the *Blame per meter traveled* by the robot. Lower the *Deviation* and *Blame*, the better. Our proposed approach C-MPDM allows the robot to modulate policy parameters appropriately so as to achieve low *Deviation* while avoiding close encounters.

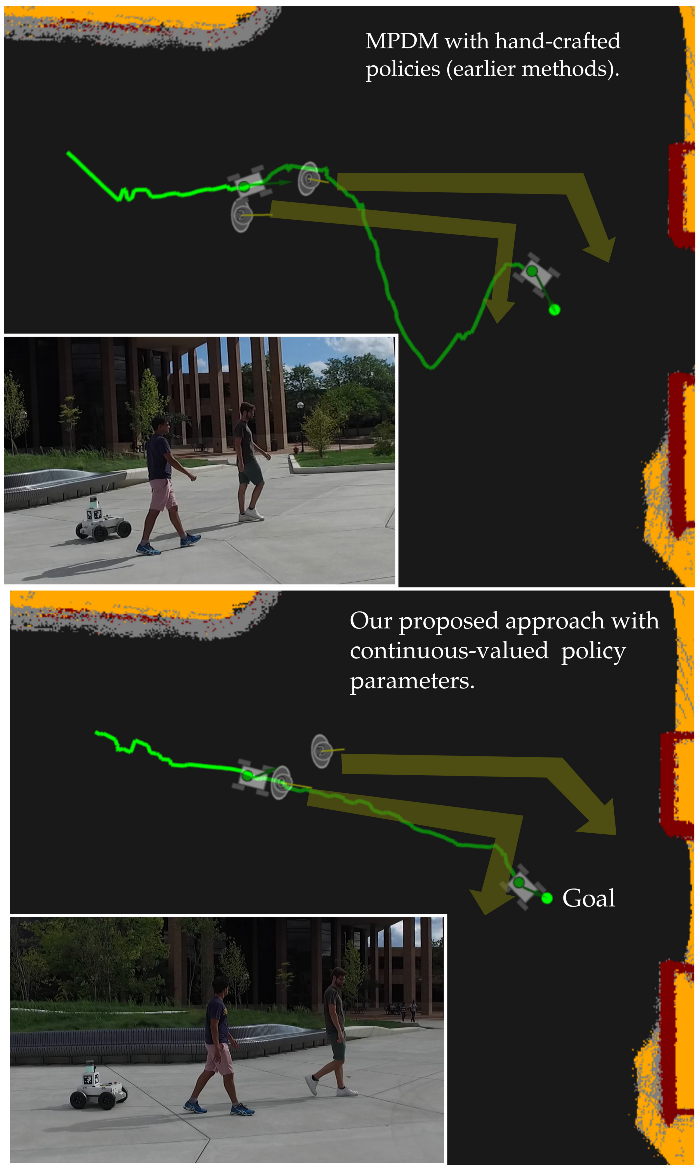

MPDM with hand-crafted policies (earlier methods).
Our proposed approach with continuous-valued policy parameters.
Goal

Figure 5.7: Robot trajectories from a real-world experiment. Two agents walk closely in front of the robot as it navigates towards its goal. The agents arbitrarily change speed and suddenly turn acutely near the robot's goal (star). The arrows mark the general trajectory of the pedestrians. Salient locations are marked by circles. The undesirable behavior from MPDM with hand-crafted policies (Π_{hc}) stems from switching between very distinct policies (*Left*). At location 1, the MPDM chooses to overtake the agents at a High speed towards the Left of the goal (see Sec. 5.4.1). However, the pedestrians also speed up slightly and the robot still lags behind as it approaches location 2 at which point, in order to make more *Progress*, the robot switches strategies and attempts to overtake from the other side (towards the Right of the goal). As the robot approaches location 3, the agents start turning acutely. and in order to reduce *Blame*, rather than going straight towards the goal, the robot tries to swerve around the agent from the Left of its goal. This extreme switch causes the robot to deviate once again. By allowing smoother transitions in nominal speed and heading, C-MPDM avoids unnecessary deviation (*Right*).

We compared C-MPDM with hand-crafted discrete policies (HC) based on the *Blame per meter traveled* as well as the trajectory *Deviation* which we define as the ratio of the extra distance traveled by the robot to the minimum (straight line) distance that the robot would travel if there were no pedestrians. Using the 10 hand-crafted policies (the planning time was insufficient to evaluate the *Follow-other* policies), the robot often finds sub-optimal policies resulting in larger trajectory *Deviation*. Our proposed method C-MPDM (purple) is able to adjust its speed and direction at a much higher resolution and as a result, finds policies that result in lower *Blame* and *Deviation*.

Fig. 5.7 shows the resultant trajectories from an experiment where two pedestrians walk closely in front of the robot, but not directly towards the robot's goal. The pedestrians arbitrarily change speed along their path and suddenly turn acutely near the robot's goal. With a small set of discrete hand-crafted policies, MPDM is forced to make extreme choices such as overtaking from the right or left, or going very slowly. Switching between these policies can result in undesirable trajectories (Fig. 5.7). C-MPDM alleviates this problem through continuous-valued parameterized policies, allowing the robot to modulate its speed and heading.

In another real-world experiment, seven volunteers were asked to move between marked points around an open space for 45 minutes. On several occasions, the volunteers were adversarial towards the robot, trying to test its capabilities by suddenly changing direction, blocking its path or jumping in front of it. We encourage the reader to see our video demonstrating emergent behavior using C-MPDM[1].

5.5 C-MPDM and MPEPC: A Comparative Study

Model Predictive Equilibrium-Point Control (MPEPC) is a state-of-the-art stochastic model predictive control algorithm where several trajectories generated from a high-fidelity robot model are quickly evaluated and the best trajectory is selected. While most other methods for safe navigation in social environments rely on slow-moving robots and effectively delegate the responsibility of avoiding collision to people, MPEPC enables safe, comfortable and customizable mobile robot navigation in dynamic social environments even at high speeds [44].

In this section, we compare the performance of C-MPDM (our proposed approach) and MPEPC on real robot platform navigating a semi-crowded, highly dynamic environment. Both algorithms were implemented on our skid-steer robot equipped with a Velodyne VLP-16 laser scanner. They share the same velocity controller and a common perception system

[1]The video is available online — https://goo.gl/WgXW55

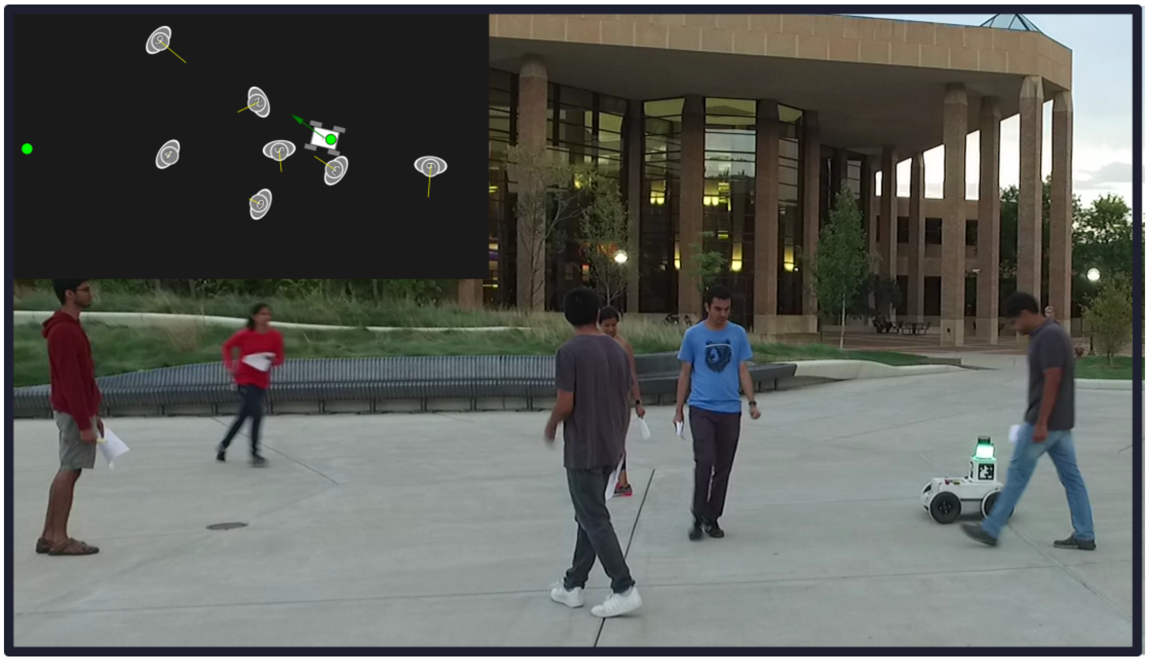


Figure 5.8: Our skid-steer robot navigating among a group of pedestrians in an open space. Both MPEPC and C-MPDM use the same perception system for localizing the robot and tracking pedestrians. *Top-Left*: GUI of the system with observed agents are depicted by gray objects. The yellow lines show the predicted trajectories for a single initial configuration of the observed agents.

for localizing the robot and tracking pedestrians.

Our operating environment is an open space, freely traversed pedestrians while the robot tries to reach a goal (Fig. 5.8). The main challenge in this environment stems from the uncertainty associated with the inferred state of the agents (people); agents can randomly slow down or come to a stop. Moreover, the complex multi-agent interactions and the unconstrained nature of this domain makes the trajectories more dependent on pedestrian configurations.

We ran structured real-world experiments, where two volunteers were asked to repeat different challenging scenarios as well as unstructured experiments, where six volunteers were asked to move spontaneously in the open space, changing speed and direction at will. Both experiments involved numerous double-blind trials, where, in each trial, the planning algorithm was chosen randomly and was unknown to the experimenter and all participants. Before discussing experimental results, the following section provides some context by highlighting key similarities and differences between MPEPC and C-MPDM.

5.5.1 Algorithmic Similarities and Differences

MPEPC is a state-of-the-art trajectory-optimization method uses receding-horizon model predictive control (MPC) to find optimal control parameters for a finite time-horizon cost function while satisfying kinodynamic constraints. Rather than optimizing over lower-level control signals (e.g. linear and angular velocities), in MPEPC, a pose-stabilizing feedback controller allows the robot to be controlled using an equilibrium target-point. This equilibrium point control (EPC) greatly reduces the dimensionality of the search-space for MPC's constrained optimization allowing MPEPC to re-plan frequently and react to sudden changes in the environment.

Both MPDM and MPEPC try to balance the robot's progress towards its goal against the inconvenience it causes to nearby pedestrians by moving too close to them. However, they differ in the possible future outcomes they consider while evaluating candidates. MPEPC considers only the most likely outcome predicted based on the current position and velocity estimates of nearby agents as well as static obstacles. The predicted trajectories for the pedestrians do not capture the complex agent-agent interactions. MPEPC incorporates uncertainty through a decaying cost function around the predicted outcome. On the other hand, in MPDM, the robot maintains a probabilistic estimate of possible initial configurations and local-policies of other agents $P(\mathbf{x}_0)$, and evaluates ego-policies by forward simulating samples drawn from this distribution. Agent-agent interactions are captured through repulsive forces in the SFM (social force model), and used for predicting future outcomes. Due to these interactions, small changes in initial configurations can result in very different outcomes. We hypothesized that MPEPC would perform worse than MPDM when agents' behaviors are coupled, since MPEPC does not model those inter-agent effects (and MPDM does).

During each planning cycle, MPEPC evaluates hundreds of candidate trajectories for a fixed time horizon T_H, each of which can be executed by a high-fidelity lower level controller. However, a high-fidelity robot model may not always be available (e.g. skid-steer systems). On the other hand, the performance of MPDM is less sensitive to the fidelity of the robot model since the planning process selects reactive policies rather than individual trajectories.

While MPEPC chooses the trajectory with the lowest cost, in C-MPDM a planning process chooses the behavioral policy with the best worst-case outcome. C-MPDM is a bilevel optimization where the upper-level optimizer (the ego-robot) chooses the policy with the most benign (low-cost) evaluation, and lower-level optimization (risk-aware policy evaluation of an ego-policy) involves finding the most potentially dangerous (likely, high-cost) outcome from all possible pedestrian configurations. As a result, MPEPC's trajectory evaluation is much faster than MPDM's policy-evaluation. MPEPC chooses locally-optimal trajectories (via an equilibrium point) every 100ms. MPDM selects locally-optimal reactive policies every 300ms, where each policy is a potential field formed by an attractive force towards a sub-goal and repulsive force away from nearby static and dynamic obstacles, which is updated on each new observation (every 20ms).

5.5.2 Structured Experiments

We present empirical results from four structured experiments described below. In each experiment, volunteers were asked to repeat a fixed scenario while the robot made its way towards its goal 15 meters away. The purpose of these structured experiments was to make multiple runs as similar as possible.

1. Two pedestrians simultaneously cross the robot orthogonally. The pedestrians also approach each other as the robot approaches them, increasing the uncertainty and the scope for agent-agent interactions.

2. The pedestrians cross the robot's trajectory obliquely, beginning from the same side of the robot. This robot must anticipate how its interaction with the one pedestrian can affect its interaction with the other pedestrian. The outcome of this scenario is sensitive to the pedestrian velocities.

3. Both pedestrians walk in front of the robot, flanking it from either side while changing direction along the way in a zig-zag pattern.

4. One pedestrian is nearly stationary in the robot's path, while the other pedestrian walks past as the robot approaches.

Each of the above scenarios was repeated 12 times and at the beginning of each trial, the robot randomly chose between MPEPC or C-MPDM. Neither the experimenter nor the pedestrian knows which algorithm is running. Only during data analysis was the algorithm revealed. We compared the performance of the trials based on the inconvenience caused to other pedestrians while the robot was in motion (*Blame per meter traveled*) as well as the time the robot took to reach its goal (*Time To Goal*).

Fig. 5.9 shows data from about 90 minutes of structured real-world experiments. In all scenarios, the robot takes roughly $5s$ more time to reach its goal when executing MPEPC as compared to C-MPDM (30% more *Time to Goal*). Furthermore, not only does C-MPDM accumulate lower *Blame* than MPEPC, but it is also more robust (lower variance in *Blame*).

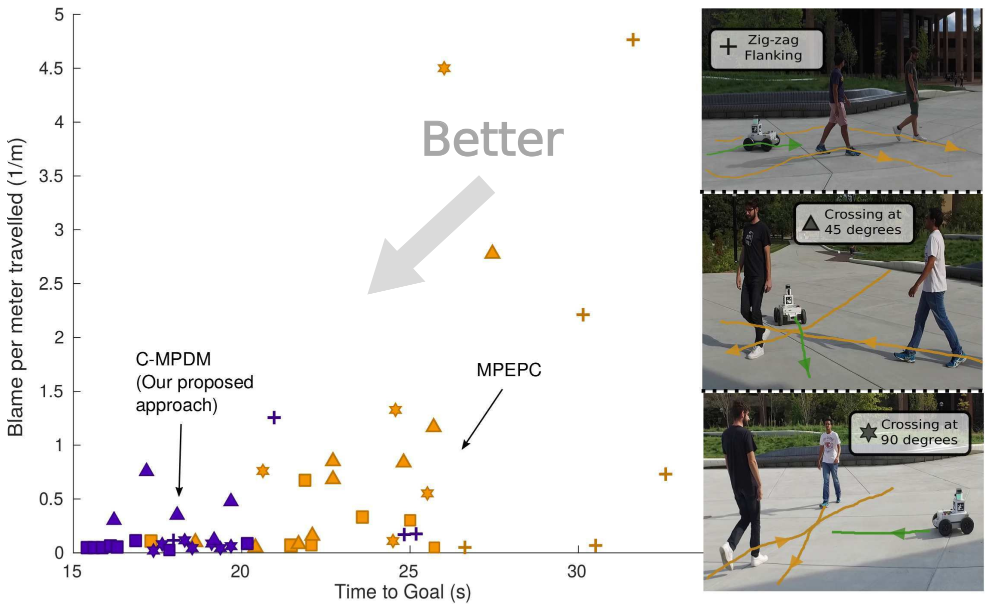
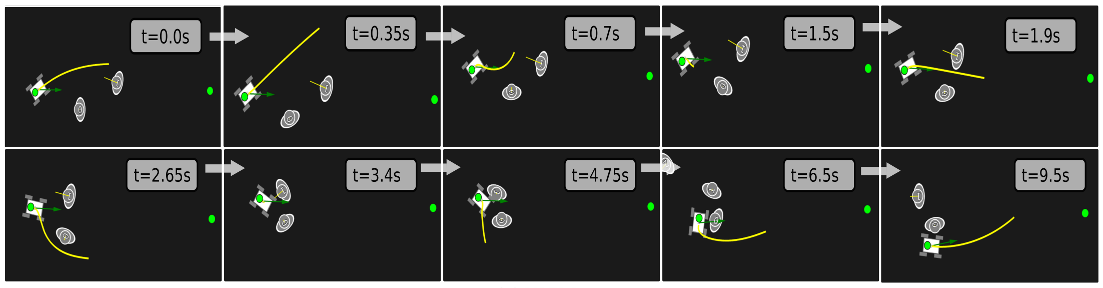


Figure 5.9: Performance of C-MPDM (purple) and MPEPC (orange) on four types of repeatable scenarios represented by the different symbols as shown in the images on the right as the robot moves towards its goal around 15 meters away. Data is collected from four repeatable experiments represented by different symbols. 1) Pedestrians crossing the robots trajectory orthogonally (star) 2) Pedestrians crossing the robot's trajectory obliquely, and beginning from the same side of the robot (Δ) 3) Pedestrians walking in front of the robot while changing direction along the way (plus) and 4) One pedestrian is nearly stationary in the robot's path, while the other pedestrian walks past when the robot approaches as shown in Fig. 5.10 (square). Each of the four experiments is repeated 12 times. For each trial, the robot chooses C-MPDM or MPEPC at random (with equal probability) and we measure the time taken for the robot to make it to its goal (*Time to Goal*) and the *Blame* per meter traveled by the robot. Lower the *Blame* and lesser the *Time to Goal*, the better. We can observe that C-MPDM is quicker than MPEPC and accumulates less *Blame* in all four experiments.

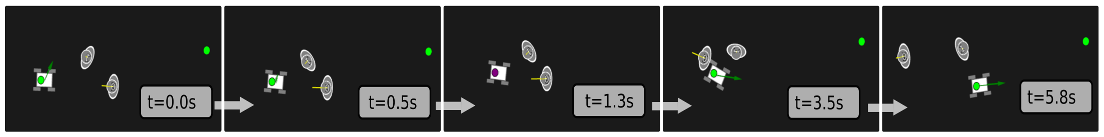
In short, in all four scenarios, we observe that the robot is quicker and more reliable when it executes C-MPDM than when it executes MPEPC.

Fig. 5.10 illustrates the difference in emergent behavior between MPEPC and MPDM through one specific scenario where one pedestrian is nearly stationary in the robot's path, while the other pedestrian walked past as the robot approached the stationary pedestrian. We notice that the optimal trajectory (in yellow) chosen by MPEPC changes dramatically in successive planning cycles, while the policies elected by C-MPDM are more consistent and result in better motion.

MPEPC assumes a simplistic constant-velocity model for the observed pedestrians. However, in this scenario, the agents' coupled interactions significantly affect the observed



(a) MPEPC



(b) C-MPDM

Figure 5.10: Sequence of plans generated by MPEPC (*Top*) and C-MDPM (*Bottom*) for a scenario where one pedestrian is nearly stationary in the robot's path, while the other pedestrian walks past when the robot approaches the stationary pedestrian. The robot is trying to reach its navigational goal (green dot). *Top*: We observe that the optimal trajectory chosen by MPEPC (yellow) can be sensitive to the pedestrian configuration. As a result, MPEPC switches between radically different 'optimal' trajectories, making little progress along any of the chosen trajectories. For example, after $1.9s$, based on the most-likely estimate of the oncoming pedestrian, MPEPC finds a trajectory between the two pedestrians. However, a slight change in direction makes this trajectory less attractive at $2.5s$. The sudden start-stop behavior between $1.5 - 2.65s$ is undesirable. *Bottom*: In C-MPDM, the planning process accounts for uncertainty in each pedestrian's current velocity as well as intentions (future speed and heading). and as a result, the elected policy is more robust to changes in the pedestrian's configuration. As the pedestrian approaches, the robot slows down at $0.5s$ and stops at $1.3s$, waiting for the pedestrian to pass before it continues towards its goal.

outcome (the pedestrian changes speed and direction on approaching the robot). Therefore, the real outcome differs significantly from the anticipated outcome due to imperfect modeling assumptions, and the optimal trajectory, being sensitive to pedestrian configuration, changes significantly in successive planning cycles. Furthermore, when MPEPC switches between radically different trajectories, our skid-steer robot's kinematic model is less accurate, undermining MPEPC's assumption of having a high-fidelity robot model. This instability in MPEPC's chosen trajectory results in the undesirable behavior shown in Fig. 5.10(a). At about $t = 2s$, a slight change in direction makes the previously chosen trajectory less attractive, resulting in a sudden start-stop behavior between $t = 1.5$ and $t = 3.5s$. The robot makes nearly no progress towards its goal for five seconds ($t = 1.5s - 6.5s$) and also inconveniences the oncoming pedestrian.

On the other hand, C-MPDM is more robust to model errors and therefore, it is more consistent in its elected policy. C-MPDM accounts for uncertainty over the current velocities and future intentions of pedestrians and the predicted outcomes capture coupled interactions between the pedestrians and the robot. Our risk-aware planner anticipates potentially dangerous future outcomes and elects the behavioral policy with the best worst-case outcome. By executing a policy rather than a full trajectory, MPDM makes fewer assumptions about the fidelity of the robot model. Fig. 5.10(b) shows that the policies elected by C-MPDM are executed for a longer duration (consistent) despite small changes in pedestrian configurations. The robot gently slows down before coming to a stop, allowing the oncoming pedestrian to pass and once its path is clear, it heads towards its goal. As a result, the oncoming pedestrian is not inconvenienced and C-MPDM saves the robot about $3.5s$ as compared to MPEPC.

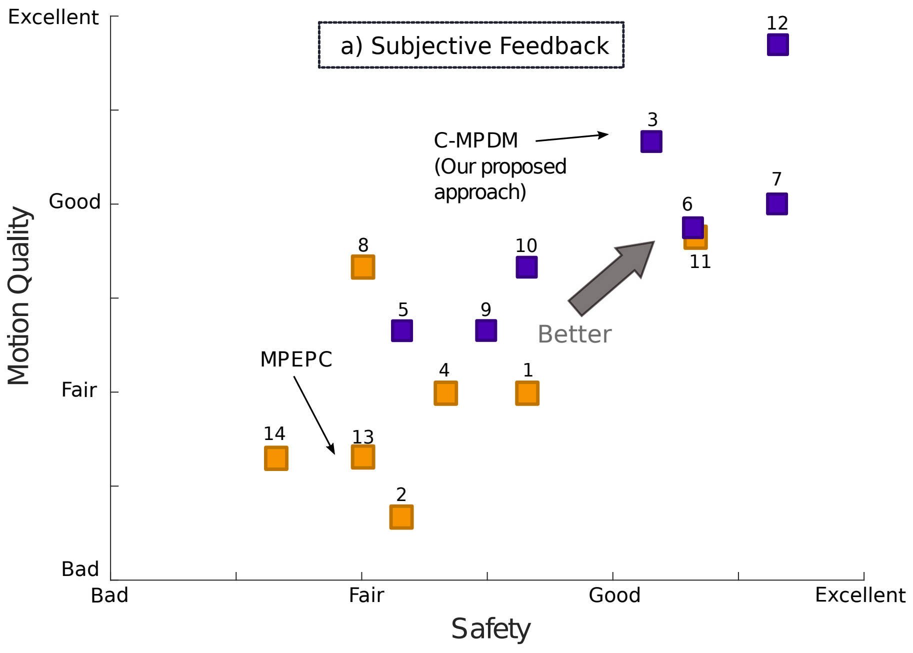
5.5.3 Randomized Double-Blind Trials

For our next experiment, we carried out fourteen trials where during each trial, six volunteers were asked to move spontaneously between markers placed around an open area as the robot moved back-and-forth between two goals 15 meters apart for 4 minutes. Volunteers were asked to change speed and direction at will. For each trial, the planner was chosen randomly (with equal probability) between C-MPDM and MPEPC. The experiment was double-blind; neither the experimenter nor the pedestrian knew which algorithm is running. After each trial, each volunteer were asked to rate (between 1 = Very Bad, 2 = Bad, 3 = Fair, 4 = Good, or 5 = Excellent) 1) how safe they felt around the robot (*Safety*) and 2) how human-like the emergent behavior of the robot was (*Motion Quality*) during the trial. Along with the subjective scores, we also record the *Blame per meter traveled* as well as the average time the robot took to reach its goal (*Average Time To Goal*) as objective metrics for each trial.

We hypothesized that because C-MPDM captures agent-agent interactions and uncertainty in pedestrian intentions directly in the decision-making process, it would outperform MPEPC in scenarios where the agents' behaviors are coupled. We test the following two hypotheses using data collected from the trials:

1. **(H1)**: C-MPDM outperforms MPEPC in the unscripted trials.
2. **(H2)**: Pedestrians prefer C-MPDM over MPEPC across the different trials.

Fig. 5.11 shows the average subjective ratings given by the volunteers as well as the objective performance for each trial. Interestingly, we observe a general agreement between the objective and subjective evaluations for each trial, regardless of the algorithm being executed. For example, according to both subjective and objective metrics, trials 6, 7,



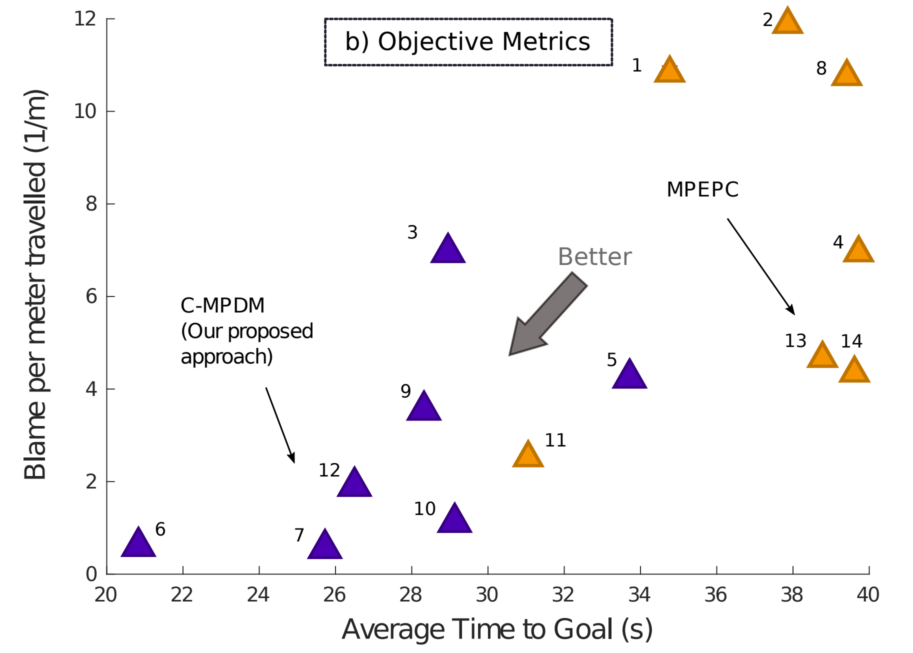


Figure 5.11: C-MPDM produces emergent behavior that is more reliable than MPEPC in dynamic uncertain environments. We measure the subjective and objective performance of the robot on fourteen double-blind trials with six volunteers in open environment. During each trial, six volunteers were asked to move in an open space as the robot moved back-and-forth between two goals 15 meters apart for 4 minutes. *Top*: Each volunteer rated the robot's *Safety* and *Motion Quality* on a scale from 1 (Very Bad) to 5 (Excellent). On most runs, volunteers felt that C-MPDM (purple squares) produced better and more reliable motion than MPEPC (orange squares). *Bottom*: The objective metrics of *Blame per meter traveled* and *Average Time to Goal* also suggest that trials with C-MPDM (purple triangles), the robot is generally safer (lower *Blame*) and almost always faster than trials with MPEPC (orange triangles).

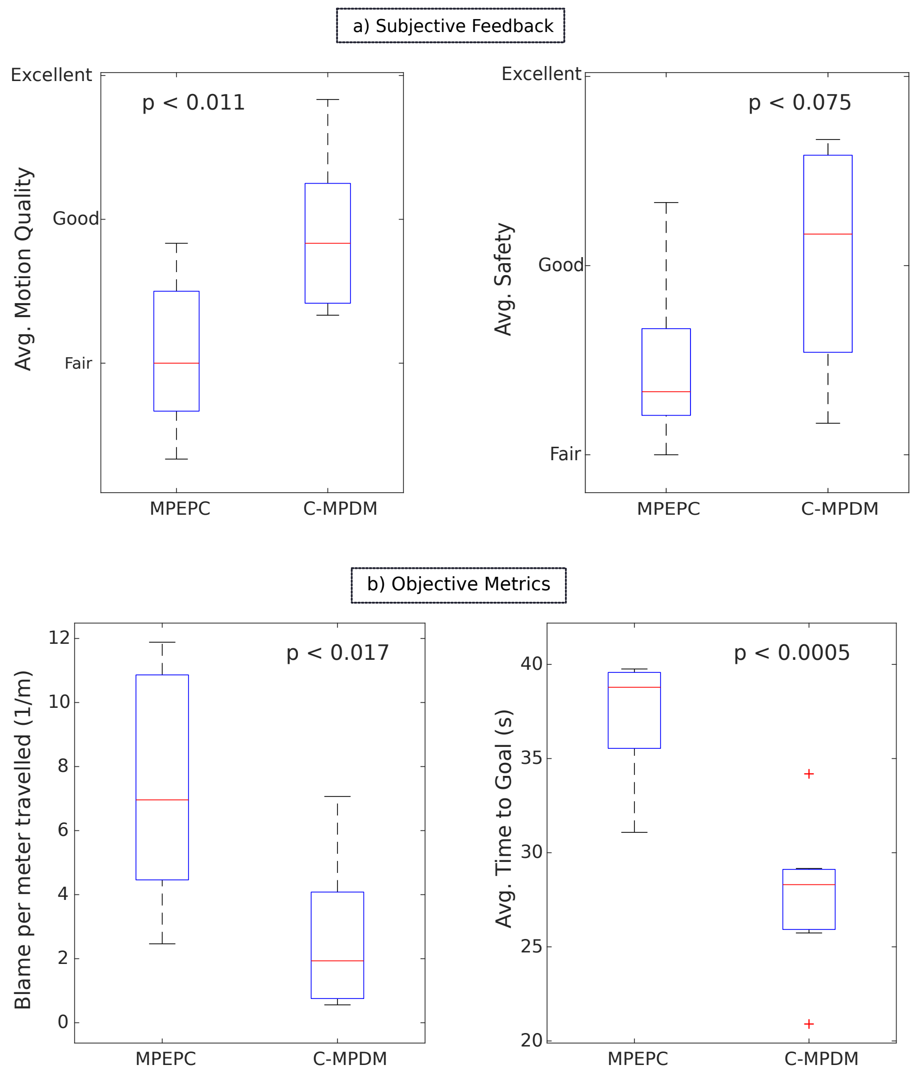


Figure 5.12: Means and confidence intervals of measured objective metrics and subjective feedback. The difference in performance across the various trials is statistically significant and in support of our first hypothesis that C-MPDM outperforms MPEPC in our trials, where the agents' behaviors are coupled. Both the objective metrics (*Blame* and *Average Time to Goal*) as well as the subjective feedback for *Motion Quality* have a p-value > 0.015. The subjective metric of *Safety* provides weaker support for due to larger variance across the trials (p-value $= 0.075$).

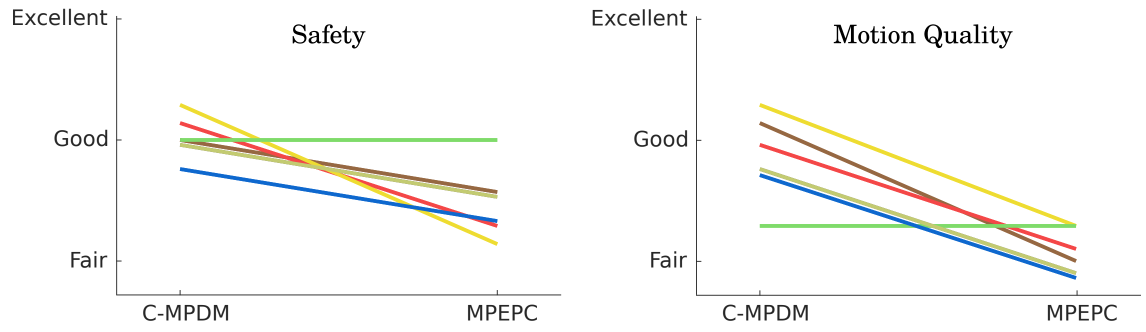


Figure 5.13: Volunteer feedback for MPEPC and C-MPDM, averaged across different trials. Each colored line-segment denotes one volunteers's ratings averaged across the trials corresponding to C-MPDM and MPEPC. The slopes of most line-segments are negative, indicating that each volunteer rated C-MPDM higher (on average) than MPEPC. Note that the experiment was double-blind and neither the experimenter nor the pedestrian knew which algorithm is running. The identity of the algorithm is only revealed after processing the data after the experiment. The difference in perceived *Motion Quality* as well as *Safety* is statistically significant, with a two-tailed p-value < 0.005 and < 0.05 respectively.

and 12 performed much better than trails 1, 2, 4, 8, 13, and 14. The objective metrics suggest that C-MPDM results in motion that is generally safer (lower *Blame*) and almost always faster than MPEPC. The subjective scores suggest that C-MPDM was generally perceived to be safer and resulted in better emergent behavior than MPEPC in our operating environment.

We used an unpaired t-test to compare the performance of both algorithms according to average subjective and objective metrics. As shown in Fig. 5.12, the difference in performance of the two algorithms across the various trials, is statistically significant in support of our first hypothesis (H1). For both objective metrics (*Blame* and *Average Time to Goal*) as well as the subjective metric of *Motion Quality* the performance difference is statistically significant with a t-value > 2.8 and a p-value > 0.015. The subjective metric of *Safety* provides weaker support for our first hypothesis due to larger variance across the trials (t-value $= 1.95$ and p-value $= 0.075$).

p-values To test our second hypothesis (H2), we average each volunteer's ratings across all trials as shown in Fig. 5.13. The different colored lines indicate the different volunteers. The non-positive slopes indicate that all of the volunteers on average perceived C-MPDM to be *Safer* and producing better *Motion Quality* than MPEPC. Using a paired sample t-test, it was determined that the difference in perceived *Safety* between the two algorithms was statistically significant with a t-value of 2.97 and a two-tailed p-value $= 0.0437$. The difference in perceived *Motion Quality* was even more significant with a t-value of 4.8 and a p-value $= 0.0049$.

After the double-blind trials, the volunteers were asked how many different algorithms were showcased by the robot during the trials. Four volunteers thought there were two systems - one that moved closer to them and was ‘harder to bully’ (C-MPDM) while the other that would stop more often and seemed ‘more confused’ (MPEPC). Two volunteers thought that there were three different systems; one in which the robot would sometimes ‘abruptly speed up in short-bursts’ (MPEPC). These testimonials show that even though both C-MPDM and MPEPC are quite safe, and effective, they are perceived differently and C-MPDM was preferred by volunteers in our experiments.

Discussion

Comparing motion planning algorithms in a dynamic social environment is challenging since navigation itself is an ill-posed problem with no objective performance metrics defining ‘good behavior’, and environmental characteristics (e.g. pedestrian density, pedestrian interactions, speed and size of the robot) heavily influence the emergent behavior. By using the same physical platform, perception sub-system and lower-level velocity controller, we ensure that C-MPDM and MPEPC differ only in their planning strategy.

We chose an outdoor environment to highlight the challenges that stem from the uncertainty associated with the inferred state of the pedestrians as well as the complex multi-agent interactions, which make future outcomes more dependent on pedestrian configurations. Through a combination of structured and unstructured experiments, as well as subjective and objective measures, we have demonstrated that by accounting for uncertainty and coupling between the agent’ behaviors and anticipating potentially dangerous future outcomes, C-MPDM is more reliable for autonomous navigation in dynamic, uncertain environments than MPEPC.

5.6 Summary

In this chapter, we have extended MPDM, allowing ego-policies to have continuous-valued parameters and reducing the need for carefully hand-engineered policies. C-MPDM radically improves the flexibility of MPDM while simultaneously satisfying real-time constraints by quickly finding promising parameters through an iterative gradient-based optimization. As a result, we can generate a continuum of risk-aware policies allowing the robot to adapt better to the dynamic environment which is critical for real-time risk-aware navigation, as demonstrated through our experimental results.

Our experiments show that for social environments, accounting for the coupled interactions of agent behaviors while predicting future outcomes becomes critical. From the large space of possible pedestrian configurations, C-MPDM uses gradient-based optimizations to focus computational resources towards finding good ego-policy parameters *now* for potentially dangerous *future* outcomes. This makes decision-making more robust to changes in human motion and tracking errors. Through extensive experiments evaluated through objective metrics and subjective feedback, we showed that C-MPDM produces emergent behavior that is more reliable than MPEPC for autonomous navigation in dynamic environments with large uncertainty.

CHAPTER 6

Conclusion

Mobile-robots have the potential to disrupt the way goods and people are transported. However, in order for autonomous robots to co-inhabit human spaces, the planning system must be able to deal with the diverse possible outcomes in everyday environments. Sensor noise and tracking errors, coupled with complex robot-agent and agent-agent interactions make it difficult to predict future outcomes. Furthermore, in dynamic environments, the robot must plan quickly in order to be able to react to sudden unexpected changes in the environment. Planning actions quickly and reliably while accounting for uncertainty is a core challenge faced by mobile robots and autonomous vehicles today. Too commonly, navigation systems don't account for agent-agent interactions while predicting future outcomes. Capturing these interactions and reasoning over them explicitly allows the robot to be simultaneously quick and reliable.

6.1 Contributions

Multi-Policy Decision Making (MPDM) is a principled framework for decision-making in environments under uncertainty with extensively coupled interactions between agents. By explicitly modeling reasonable behaviors of both the robot and other agents' policies, we make informed high-level behavioral decisions that account for the consequences of the ego-robot's actions. MPDM is a hybrid navigation system where a slower policy election anticipates potentially influential future outcomes, captures pedestrian intentions and interactions and discovers effective ego-policies. The chosen reactive policy is then executed by the lower-level controller and is responsible for basic obstacle avoidance.

In chapter 3, we propose MPDM as 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) [40] 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. We demonstrate the robustness of switching between higher-level behaviors to sensor noise and study the effect of the conservatism of the perception module's state estimator through simulation experiments.

Our approach works well for navigating among pedestrians in an indoor hallway, where the possible future outcomes are constrained by the two ends of the corridor. However, open spaces present a much larger space of possible outcomes since pedestrians can change direction at will. With increased uncertainty, we found that traditional MPDM methods often underestimated the inconvenience caused to pedestrians while evaluating its candidate policies, resulting in poor decision making.

Our next contribution resolves the following question: Can we reliably evaluate ego-policies with few samples given the large space of possible outcomes? The key challenge to reliable policy evaluation arises from the uncertainty associated with the inferred state of the agents (people) and the complex multi-agent interactions which make the forward-simulated trajectories sensitive to the initial configurations sampled. In our application, it is especially difficult to sample bad outcomes because we assume that all agents follow policies that tend to avoid collisions and dangerous scenarios in the first place. Sampling randomly is likely to miss high-cost events, even if they are individually reasonably probable (high probability density) because of the scarcity of such configurations in the state space (low total probability mass of high-cost outcomes).

Rather than rely on random sampling, in chapter 4, we reformulate policy evaluation, biasing sampling towards increasingly likely and high-cost outcomes. We propose an anytime algorithm for finding increasingly influential outcomes through a gradient-based optimization where accurate gradients are computed efficiently by representing a forward simulation as a deep network and enabling effective backpropagation. Through extensive simulation as well as real-world experiments, we demonstrate a dramatic improvement in reliability for the same number of candidate ego-policies being evaluated in real-time as before.

Next, we resolve a core tension in MPDM type systems — it is desirable to add more policies to the system to increase the expressivity of the system, however, this increases computational cost. Chapter 5 achieves expressivity in a different way than previous MPDM approaches: it allows policies to have one or more *continuous* parameters, and then efficiently computes good values of those continuous parameters. In this way, MPDM no longer requires a set of carefully hand-crafted policies. Rather, we are able to generate promising context-specific ego-policies in real-time. Overall, this thesis paints the navigation problem in a new light, transforming a POMDP into a Stackelberg game or a bilevel optimization.

6.2 Future Work

Multi-Policy Decision Making (MPDM) is a principled framework for decision-making in environments under uncertainty where forward simulations capture coupled interactions between agents' behaviors. The coupled intent estimation and behavioral planning framework leaves a lot of flexibility for the system designer. We believe that the ideas of representing forward simulations as deep networks and computing gradients to discover influential agent configurations, as well as promising ego-policy parameters, can serve as a general tool for coupled prediction and planning under uncertainty. In the remainder of this section, we discuss some interesting extensions of MPDM.

6.2.1 Learning Policies

Rather than relying on human-designed local planning algorithms, the policies in MPDM could be learned from data. One of the fundamental limitations of end-to-end learning is that it is difficult to get the data, especially since actions change future observations. However, learning local policies such as following a pedestrian or coming to a stop gracefully can generalize better with less data [67, 157, 158].

Multi-policy decision making provides a unique opportunity to combine these policies from different sources for effective navigation. Our recurrent-network based representation of a forward simulation allows learned policies to be directly incorporated into the MPDM framework; the motion model from Fig. 4.3 would have to be replaced by the learned policy network.

6.2.2 Parallelizing MPDM for Multi-Modal Uncertainty

The gradient-based approaches in this thesis discover local optima or local saddle points. In more complex real-world applications such as autonomous driving, in addition to the cost function, the posterior distribution $P(\boldsymbol{x}_0)$ may also be multi-modal (a vehicle may yield at a round-about, or merge with the traffic). With a multi-modal objective function, the number of samples needed to reliably evaluate a policy grows with the number of modes. Additionally, if the ego-robot has several distinct parameterized policies, the computational requirement would correspondingly increase.

Fortunately, parallelizing MPDM can help scale the framework to more complex, real-world domains. Not only can each ego-policy be evaluated independently, but the core optimization algorithms can be run in parallel for different seed initial configurations. Furthermore, the computation of accurate gradients through backpropagation can be parallelized wherever possible.

6.2.3 MPDM with Explicit Communication

To date, MPDM policies have used only *implicit* communication– the information that is exchanged between agents is limited to the physically visible actions that they take. There are several situations where which might be difficult to get around by motion planning, where ad-hoc communication might be tremendously useful. Honking is a good example, where an existing “alphabet” of communication symbols (honking and not honking) are already established and the meaning of those symbols–even if ambiguous– is not entirely under our control.

Honking is a communication channel used by human drivers which communicates not only state but also can influence behavior by drawing attention (“watch out, I’m here!”). The main challenge is in modeling how other agents will react to such communication. For instance, if acknowledged, honking can change an agent’s behavior, but it is not clear how. Fortunately, MPDM is amenable to planning under uncertainty. Such an approach would require us to find a set of policies that, when assigned to other agents, predict their behavior both when a nearby agent is honking and when not. This is not fundamentally different than what we already must do in MPDM, though collecting more data of agents interacting while honking will be necessary in order for us to measure the accuracy of our modeled policies.

BIBLIOGRAPHY

[1] A. I. Center, "Shakey the robot," 1984.

[2] J. Enright and P. R. Wurman, "Optimization and coordinated autonomy in mobile fulfillment systems." 2011.

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

[4] J. A. Fodor, "The modularity of mind," 1983.

[5] I. Goldstein and G. G. Hendrix, "The role of representation in artificial intelligence(tutorial session)," in *Proceedings of the 1976 Annual Conference*, ser. ACM '76. New York, NY, USA: ACM, 1976, pp. 70–72, chairman-Fikes, Richard. [Online]. Available: http://doi.acm.org/10.1145/800191.805529

[6] R. A. Brooks, "Intelligence without representation," *Artificial intelligence*, vol. 47, no. 1-3, pp. 139–159, 1991.

[7] R. C. Arkin, R. C. Arkin, *et al.*, *Behavior-based robotics*, 1998.

[8] 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.

[9] 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.

[10] J. J. Park, C. Johnson, and B. Kuipers, "Robot navigation with model predictive equilibrium point control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4945–4952.

[11] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1772–1780.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] V. Braitenberg, *Vehicles, experiments in synthetic psychology.* MIT Press, 1984.

[17] H. A. Simon, *The sciences of the artificial*, 1996.

[18] R. Brooks, "A robust layered control system for a mobile robot," *IEEE journal on robotics and automation*, vol. 2, no. 1, pp. 14–23, 1986.

[19] M. Wooldridge, *An introduction to multiagent systems.* John Wiley & Sons, 2009.

[20] D. Payton, "An architecture for reflexive autonomous vehicle control," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3. IEEE, 1986, pp. 1838–1845.

[21] A. Saffiotti, K. Konolige, and E. H. Ruspini, "A multivalued logic approach to integrating planning and control," *Artificial intelligence*, vol. 76, no. 1-2, pp. 481–526, 1995.

[22] R. C. Arkin, "Motor schemabased mobile robot navigation," *The International journal of robotics research*, vol. 8, no. 4, pp. 92–112, 1989.

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

[24] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 500–505.

[25] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on.* IEEE, 1990, pp. 572–577.

[26] 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.

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

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

[29] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 1190–1197.

[30] P. Maes, "Situated agents can have goals." 1990.

[31] J. K. Rosenblatt and D. Payton, "A fine-grained alternative to the subsumption architecture for mobile robot control."

[32] J. K. Rosenblatt, "Damn: A distributed architecture for mobile navigation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 339–360, 1997.

[33] S. Hanks and R. J. Firby, "Issues and architectures for planning and execution," in *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*. Morgan Kaufmann, 1990, pp. 59–70.

[34] M. J. Mataric, "Behaviour-based control: Examples from navigation, learning, and group behaviour," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 323–336, 1997.

[35] M. J. Matarić and F. Michaud, "Behavior-based systems," in *Springer Handbook of Robotics*. Springer, 2008, pp. 891–909.

[36] J. H. Connell, "Sss: A hybrid architecture applied to robot navigation," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 2719–2724.

[37] R. C. Arkin, "Towards the unification of navigational planning and reactive control," 1989.

[38] B. Pell, D. E. Bernard, S. A. Chien, E. Gat, N. Muscettola, P. P. Nayak, M. D. Wagner, and B. C. Williams, "An autonomous spacecraft agent prototype," *Autonomous Robots*, vol. 5, no. 1, pp. 29–52, 1998.

[39] E. Gat, "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots."

[40] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, May 2015.

[41] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, vol. 41, no. 6, pp. 1367–1382, August 2017.

[42] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard, "A bilevel model for toll optimization on a multicommodity transportation network," *Transportation Science*, vol. 35, no. 4, pp. 345–358, 2001.

[43] C. Jones and M. Morari, "Approximate explicit mpc using bilevel optimization," in *Control Conference (ECC), 2009 European*. IEEE, 2009, pp. 2396–2401.

[44] J. J. Park, "Graceful navigation for mobile robots in dynamic and uncertain environments." 2016.

[45] C. Johnson and B. Kuipers, "Socially-aware navigation using topological maps and social norm learning," 2018.

[46] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[47] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 749–765.

[48] D. M. Wolpert and Z. Ghahramani, "Computational principles of movement neuroscience," *Nature neuroscience*, vol. 3, no. 11s, p. 1212, 2000.

[49] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, Sept. 2008, pp. 1056–1062.

[50] N. Du Toit and J. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, USA, May 2010, pp. 966–973.

[51] D. Vasquez, T. Fraichard, and C. Laugier, "Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1486–1506, 2009.

[52] A. D. V. Govea, *Incremental learning for motion prediction of pedestrians and vehicles*. Springer Science & Business Media, 2010, vol. 64.

[53] I. Pérez-Hurtado, J. Capitán, F. Caballero, and L. Merino, "An extension of ghmms for environments with occlusions and automatic goal discovery for person trajectory prediction," in *Mobile Robots (ECMR), 2015 European Conference on*. IEEE, 2015, pp. 1–7.

[54] M. W. Turek, A. Hoogs, and R. Collins, "Unsupervised learning of functional categories in video scenes," in *European Conference on Computer Vision*. Springer, 2010, pp. 664–677.

[55] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2203–2210.

[56] S. Yi, H. Li, and X. Wang, "Understanding pedestrian behaviors from stationary crowd groups," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3488–3496.

[57] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "An optimality principle governing human walking," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 5–14, 2008.

[58] A. Bera, T. Randhavane, R. Prinja, and D. Manocha, "Sociosense: Robot navigation amongst pedestrians with social and psychological constraints," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 7018–7025.

[59] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision*. Springer, 2012, pp. 201–214.

[60] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection," *Neural Networks*, 2018.

[61] T. Fernando, S. Denman, A. McFadyen, S. Sridharan, and C. Fookes, "Tree memory networks for modelling long-term temporal dependencies," *Neurocomputing*, vol. 304, pp. 64–81, 2018.

[62] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," *arXiv preprint arXiv:1705.02503*, 2017.

[63] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[64] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[65] R. Hug, S. Becker, W. Hübner, and M. Arens, "On the reliability of lstm-mdl models for pedestrian trajectory prediction."

[66] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, and D. Prattichizzo, "When Helbing meets Laumond: the headed social force model," in *IEEE Conference on Decision and Control (CDC)*, 2016, pp. 3548–3553.

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

[68] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, AB, Canada, Aug. 2005, pp. 2210–2215.

[69] T. Ohki, K. Nagatani, and K. Yoshida, "Collision avoidance method for mobile robot considering motion and personal spaces of evacuees," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 1819–1824.

[70] J. Choi, G. Eoh, J. Kim, Y. Yoon, J. Park, and B.-H. Lee, "Analytic collision anticipation technology considering agents' future behavior," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 1656–1661.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

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

[77] D. Ferguson, T. M. Howard, and M. Likhachev, “Motion planning in urban environments,” *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[78] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenet frame,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, USA, May 2010, pp. 987–993.

[79] W. Xu, J. Wei, J. Dolan, H. Zhao, and H. Zha, “A real-time motion planner with trajectory optimization for autonomous vehicles,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, May 2012, pp. 2061–2067.

[80] J. Hardy and M. Campbell, “Contingency planning over probabilistic obstacle predictions for autonomous road vehicles,” *IEEE Transactions of Robotics*, vol. 29, no. 4, pp. 913–929, 2013.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] T. Erez, Y. Tassa, and E. Todorov, “Infinite-horizon model predictive control for periodic tasks with contacts,” *Robotics: Science and systems VII*, p. 73, 2012.

[86] J. H. Lee, “Model predictive control: Review of the three decades of development,” *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.

[87] P. Trautman and A. Krause, “Unfreezing the robot: Navigation in dense, interacting crowds,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 797–803.

[88] K. Kim, D. Lee, and I. Essa, “Gaussian process regression flow for analysis of motion trajectories,” in *Proceedings of the IEEE International Conference on Computer Vision*, Barcelona, Spain, Nov. 2011, pp. 1164–1171.

[89] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, “A Bayesian nonparametric approach to modeling motion patterns,” *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[90] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns,” *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[91] Q. Tran and J. Firl, “Modelling of traffic situations at urban intersections with probabilistic non-parametric regression,” in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Gold Coast City, Australia, June 2013, pp. 334–339.

[92] ——, “Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression,” in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, MI, USA, June 2014, pp. 918–923.

[93] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, “Apprenticeship learning for motion planning with application to parking lot navigation,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1083–1090.

[94] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, “Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior,” in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 322–331.

[95] P. Henry, C. Vollmer, B. Ferris, and D. Fox, “Learning to navigate through crowded environments,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 981–986.

[96] B. Kim and J. Pineau, “Socially adaptive path planning in human environments using inverse reinforcement learning,” *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[97] B. Okal and K. O. Arras, “Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2889–2895.

[98] M. Kuderer, S. Gulati, and W. Burgard, “Learning driving styles for autonomous vehicles from demonstration,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015, pp. 2641–2646.

[99] L. Ran, Y. Zhang, Q. Zhang, and T. Yang, “Convolutional neural network-based robot navigation using uncalibrated spherical images,” *Sensors*, vol. 17, no. 6, p. 1341, 2017.

[100] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in *Intelligent Robots and*

Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017, pp. 31–36.

[101] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[102] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

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

[104] L. Tai, J. Zhang, M. Liu, J. Boedecker, and W. Burgard, "A survey of deep network solutions for learning control in robotics: From reinforcement to imitation," *arXiv preprint arXiv:1612.07139*, 2016.

[105] DARPA, "DARPA Urban Challenge," http://archive.darpa.mil/grandchallenge/, 2007.

[106] M. Montemerlo *et al.*, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[107] I. Miller *et al.*, "Team Cornell's Skynet: Robust perception and planning in an urban environment," *Journal of Field Robotics*, vol. 25, no. 8, pp. 493–527, 2008.

[108] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. . Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[109] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[110] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and related stochastic optimization problems," *Artificial Intelligence*, vol. 147, no. 1–2, pp. 5–34, 2003.

[111] S. Thrun, "Monte Carlo POMDPs," *Proceedings of the Advances in Neural Information Processing Systems Conference*, pp. 1064–1070, 2000.

[112] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proceedings of the Robotics: Science & Systems Conference*, Zurich, Switzerland, June 2008.

[113] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2164–2172.

[114] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[115] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based MDP for robust single-lane autonomous driving behavior under uncertainties," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2586–2592.

[116] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic MDP-behavior planning for cars," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2011, pp. 1537–1542.

[117] S. Candido, J. Davidson, and S. Hutchinson, "Exploiting domain knowledge in planning for uncertain robot systems modeled as pomdps," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, USA, May 2010, pp. 3596–3603.

[118] T. Lee and Y. J. Kim, "Massively parallel motion planning algorithms under uncertainty using POMDP," *International Journal of Robotics Research*, vol. 35, no. 8, pp. 928–942, 2016.

[119] R. He, E. Brunskill, and N. Roy, "Efficient planning under uncertainty with macro-actions," *Journal of Artificial Intelligence Research*, vol. 40, pp. 523–570, 2011.

[120] T. Bandyopadhyay, K. Won, E. Frazzoli, D. Hsu, W. Lee, and D. Rus, "Intention-aware motion planning," in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, ser. Springer Tracts in Advanced Robotics, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds. Springer Berlin Heidelberg, 2013, vol. 86, pp. 475–491.

[121] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2014, pp. 392–399.

[122] N. Ye, A. Somani, D. Hsu, and W. Lee, "DESPOT: Online POMDP planning with regularization," vol. 58, pp. 231–266, 2017.

[123] Y. Luo, H. Bai, D. Hsu, and W. S. Lee, "Importance sampling for online planning under uncertainty," in *Workshop on Algorithmic Foundations of Robotics*, 2016.

[124] H. Bai, S. Cai, D. Hsu, and W. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2015.

[125] G. Ferrer, A. Garrell, F. Herrero, and A. Sanfeliu, "Robot social-aware navigation framework to accompany people walking side-by-side," *Autonomous Robots*, pp. 1–19, 2016.

[126] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 454–460.

[127] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, pp. 15–31, 2016.

[128] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction," in *Proceedings of the Robotics: Science & Systems Conference*, Rome, Italy, July 2015.

[129] ——, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, pp. 1–16, 2017.

[130] D. A. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, 1989.

[131] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems: a survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[132] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[133] H. von Stackelberg, *The Theory of the Market Economy*. Oxford University Press, 1952.

[134] A. Sinha, P. Malo, A. Frantsev, and K. Deb, "Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 478–485.

[135] A. U. Raghunathan and L. T. Biegler, "Mathematical programs with equilibrium constraints (mpecs) in process engineering," *Computers & chemical engineering*, vol. 27, no. 10, pp. 1381–1392, 2003.

[136] H. I. Calvete, C. Galé, and M.-J. Oliveros, "Bilevel model for production–distribution planning solved by using ant colony optimization," *Computers & operations research*, vol. 38, no. 1, pp. 320–327, 2011.

[137] P. Hansen, B. Jaumard, and G. Savard, "New branch-and-bound rules for linear bilevel programming," *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 5, pp. 1194–1217, 1992.

[138] L. Vicente, G. Savard, and J. Júdice, "Descent approaches for quadratic bilevel programming," *Journal of Optimization Theory and Applications*, vol. 81, no. 2, pp. 379–399, 1994.

[139] K. J. Arrow, L. Hurwicz, H. Uzawa, and H. B. Chenery, "Studies in linear and nonlinear programming," 1958.

[140] A. Cherukuri, B. Gharesifard, and J. Cortes, "Saddle-point dynamics: conditions for asymptotic stability of saddle points," *SIAM Journal on Control and Optimization*, vol. 55, no. 1, pp. 486–511, 2017.

[141] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 3800–3805.

[142] D. Richert and J. Cortés, "Robust distributed linear programming," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2567–2582, 2015.

[143] X. Zhang and A. Papachristodoulou, "A real-time control framework for smart power networks with star topology," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 5062–5067.

[144] 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.

[145] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on*. IEEE, 2010, pp. 4057–4062.

[146] S. M. Ross, *A course in simulation*. Prentice Hall PTR, 1990.

[147] R. Korn, E. Korn, and G. Kroisandt, *Monte Carlo methods and models in finance and insurance*. CRC press, 2010.

[148] D. Mehta, G. Ferrer, and E. Olson, "Fast discovery of influential outcomes for risk-aware MPDM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017, https://april.eecs.umich.edu/papers/details.php?name=mehta2017icra.

[149] ——, "Backprop-MPDM: Faster risk-aware policy evaluation through efficient gradient optimization," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2018.

[150] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of operations research*, vol. 153, no. 1, pp. 235–256, 2007.

[151] J. P. Hanna, P. S. Thomas, P. Stone, and S. Niekum, "Data-efficient policy evaluation through behavior policy search," *arXiv preprint arXiv:1706.03469*, 2017.

[152] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[153] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[154] R. F. Stengel, *Optimal control and estimation*. Courier Corporation, 2012.

[155] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the American Control Conference*. IEEE, 2005, pp. 300–306.

[156] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-Trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[157] R. Goeddel, "Policy-based planning for robust robot navigation," Ph.D. dissertation, University of Michigan, Ann Arbor, USA, October, 2018.

[158] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.