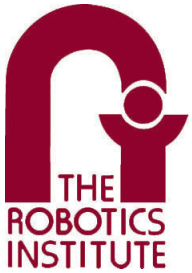
Safe and Efficient Navigation in Dynamic Environments

Anirudh Vemula

CMU-RI-TR-17-40

July 2017



The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Supervisors:
Dr. Jean Oh
Dr. Katharina Muelling

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

For my brother, who put the fear of mediocrity in me.
For my mom, for all her sacrifices to help me pursue my dreams.

Abstract

For mobile robots to become ubiquitous, they need to be able to navigate in dynamic environments in a safe and efficient way. This is a challenging problem due to the added time dimension in the search space and the subtle interactions between dynamic agents that are extremely difficult to model. In this thesis, we will address both of these challenges.

The challenge of dimensionality is addressed by proposing a novel path planning algorithm in environments with dynamic agents with quick planning times. We apply the idea of adaptive dimensionality to speed up path planning in dynamic environments for a robot with no assumptions on its dynamic model. Specifically, our approach considers the time dimension only in those regions of the environment where a potential collision may occur, and plans in a low-dimensional state-space elsewhere. We show that our approach is complete and is guaranteed to find a solution, if one exists, within a cost sub-optimality bound. We experimentally validate our method on the problem of 3D nonholonomic vehicle navigation in dynamic environments. Our results show that the presented approach achieves substantial speedups in planning time over 4D heuristic-based A*, especially when the resulting plan deviates significantly from the one suggested by the heuristic.

We tackle the challenge of modeling interactions by presenting a novel statistical model to capture cooperative behavior in human crowds. Previous approaches have used handcrafted functions based on proximity to model human-human and human-robot interactions. However, these approaches can only model simple interactions and fail to generalize for complex crowded settings. We develop an approach that models the joint distribution over future trajectories of all interacting agents in the crowd through a local interaction model that we train using real human trajectory data. The interaction model infers the velocity of each agent based on the spatial orientation of other agents in his vicinity. During prediction, our approach infers the goal of the agent from its past trajectory and uses the learned model to predict its future trajectory. We demonstrate the performance of our method against a state-of-the-art approach on a public dataset and show that our model outperforms when predicting future trajectories for longer horizons.

Finally, we lay out future directions of research in the domain of robot navigation in dynamic environments, and the challenges remaining. We plan to verify and validate the proposed work in this thesis on a robot placed in a real human crowd. Other challenges include more accurate long-term prediction, uncertainty associated with predictions and real-time incremental planning algorithms.

Acknowledgements

I would like to thank my advisors, Jean Oh and Katharina Muelling for their mentorship and guidance for the past two years. I am extremely grateful to them for giving me the freedom to pursue research topics that I am excited about and provide their valuable insights to help me achieve my goals.

I would also like to thank a bunch of people for their technical assistance in several stages of my research: Pete Trautman for providing useful insights into a difficult problem, Venkatraman Narayanan for demistifying graph search and Kalin Gochev for going through the painstaking process of explaining SBPL code. I am also indebted to Sanjiban Choudhury for his constant guidance, since my days as an intern, in my research directions and for his thoughtful opinions.

A big shout out to my colleagues at the Robotics Institute, Puneet Puri, Vishal Dugar, Debidatta Dwibedi, Rosario Scalise, Sankalp Arora, Jerry Hsiung, Achal Dave, and many others for lending an ear when it was most needed. A special thanks goes to Puneet for the brainstorming sessions regarding our research, which helped me gain a better perspective.

Finally, I would like to thank my brother for pushing me to strive for the best and not settle for less, and my parents for their unwavering support of my choices in life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will motivate the problem tackled by this thesis and highlight the challenges involved. We will then present necessary background on the problems of path planning and trajectory prediction. We finish the chapter by describing the organization of the thesis.

1.1 Planning in Dynamic Environments

Navigation is an essential task for the autonomous mobile robot. The task not only involves how the robot can move on its own, but also how to achieve various goals and objectives. Specifically, the robot has to achieve its goals in the presence of hard constraints such as dynamic feasibility, collision avoidance, and soft constraints like social compliance. These problems involve finding trajectories in the workspace from a start configuration to a goal configuration that satisfy the aforementioned constraints. *Path Planning* is the problem of finding such a trajectory. The path planning approach depends on two important aspects of the planner: The agent needs to have an accurate model of the dynamics of the environment or state of the world around it, and it needs to have a model of how its actions affect the world state. For mobile robots, this is significantly challenging as they have to rely on perception to create a model of the world state. In the absence of an accurate long-horizon models and limited perception, mobile robots were traditionally relegated to use reactive methods or one-step planning algorithms, [2, 3]. These reactive methods map the observed state of the world at the current time-step to an action for the robot to execute. Employing an accurate model of the environment, the mobile robot can instead plan its actions ahead, to achieve more efficient (in some cases, more social) behavior. However, the major challenge in achieving this objective is modeling the state of the environment, which can be dynamic.

Dynamics is an important aspect of mobile robot navigation in partially known or completely unknown environments. There are two types of dynamics that need to be considered when we perform planning: the first are the kinodynamic constraints of the robot itself, the second are the dynamics of other agents in the environment. Most modern planning algorithms consider the kinodynamic constraints of the robot to come up with feasible trajectories for the robot to execute. To ensure that the resulting trajectory doesn't collide with other agents, we need to come up with an accurate model of the environment. Such a model enables accurate long-term predictions of the trajectories of other agents in the

environment. The dynamics of the environment are not readily predictable and need to be supplemented by sensory information. Given sensory information regarding the past behavior of the agents, the model needs to predict their future behavior.

We envision mobile robots to coexist with humans in unscripted environments and accomplish various kinds of tasks. Existing works towards this goal started out in the 1990s with RHINO, [4], and MINERVA, [5], both designed as interactive museum tour guide robots that will move among humans in museums. These works made extensive use of path planning, localization and mapping methods that were developed around the same time, and pioneered the field of service robots inciting a period of exciting research in the area of robot navigation in human crowds (see Chapter 2 for a more detailed literature review).

Despite the decades of research towards this goal, there are several fundamental problems that remain unsolved. Traditional algorithms for navigation in dynamic environments tend to make strong assumptions about the environment and its own motion model, in order to efficiently plan paths quickly [6, 7]. These planning algorithms don't extend readily to unscripted environments where the agent is highly uncertain about the dynamics of the environment. Most of the work towards modeling dynamics of an unknown environment make simplifying reductions such as independence of motion and the number of surrounding agents [8]. Situations where the actions of each agent affect each other are not modeled. This is especially true in the case of human-robot interactions. To make the vision of coexisting robots and humans a reality, there is a great need for efficient planning algorithms in dynamic environments and accurate models for environment dynamics.

In this thesis, we focus on the problems of planning and modeling dynamics. More specifically, the first part of the thesis deals with path planning in dynamic environments, with known dynamic model of the environment, and the latter half deals with human trajectory prediction in dense crowds. We present novel solutions and take a small step towards making the goal of autonomous mobile robot navigation more realizable.

1.2 Problem Definition

In this section, we describe the formal problem definitions for the path planning and trajectory prediction problems. Subsequently, we describe how planning can be reduced to inference in a joint prediction model and briefly present the *freezing robot problem*, which was first introduced in Trautman and Krause [1].

1.2.1 Path Planning Problem in Dynamic Environments

The general path planning problem for an explicit goal configuration is, given the robot dynamics, its start configuration and the configuration space, to find a path that is collision-free, dynamically feasible to execute and is a solution.

Let $X \subset \mathbb{R}^n$ be the configuration space of the system. Let $\mathcal{T} = [0, 1]$ be the time interval of interest without loss of generality. We define *configuration-time space* by incorporating time as an additional dimension in the configuration space, formed as $X \times \mathcal{T}$ and denoted by $X\mathcal{T}$. It consists of pairs $\langle x, t \rangle$, where x is an element of X denoting the robot's configuration, and t is a scalar belonging to $\mathcal{T}$ denoting time. Let $X\mathcal{T}_{obs} \subset X\mathcal{T}$ be the set of all pairs $\langle x, t \rangle$ such that a robot configured at x at time t collides with any moving (or stationary) obstacle. The dynamics of the system is specified as a dynamics constraint $g(x, \frac{dx}{dt}, \ldots, \frac{d^r x}{dt^r}) \leq 0, x \in$

X. Let the trajectory $\xi : [0, 1] \rightarrow X$ be a continuous mapping from time to configuration. The planning problem is to find the shortest dynamically feasible trajectory from start x_0 to the goal x_f that is collision free. This is expressed as follows:

$$\begin{aligned} \underset{\xi(t)}{\text{minimize}} \quad & \int_0^1 \|\dot{\xi}(t)\| \mathrm{dt} \\ \text{subject to} \quad & \xi(0) = x_0 \\ & \xi(1) = x_f \\ & g(\xi, \frac{d\xi}{dt}, \ldots, \frac{d^r\xi}{dt^r}) \leq 0 \\ & \langle \xi(t), t \rangle \in X\mathcal{T} \setminus X\mathcal{T}_{obs}, t \in [0, 1]. \end{aligned} \tag{1.1}$$

1.2.2 Trajectory Prediction Problem

Trajectory prediction in a dynamic environment entails predicting the future trajectories of all dynamic agents in the environment until a fixed horizon. As argued before, this problem is challenging as we need to model the world dynamics to obtain accurate predictions.

Given a dynamic environment with agents indexed by the set $\{1, 2, \cdots, N\}$, where N is the number of agents in the environment. The trajectory of agent i is given by $\mathbf{f}^{(i)} = (f_1^{(i)}, f_2^{(i)}, \cdots, f_T^{(i)})$ where T is the length of trajectory, and $f_t^{(i)}$ represents the location of agent i at time t. We also assume that we have access to their past observed locations given by $\mathbf{z}$, where $\mathbf{z}^{(i)} = (z_1^{(i)}, z_2^{(i)}, \cdots, z_t^{(i)})$ are the observed locations of agent i until time t.

The trajectory prediction problem is to model the following joint distribution

$$P(\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \cdots, \mathbf{f}^{(N)} | \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(N)}), \tag{1.2}$$

i.e., given their observed locations $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(N)}$, we want to estimate the distribution over their trajectories until some future time T. Note that such joint modeling accounts for dependencies between trajectories of different agents which, as we show later, is paramount for accurate predictions.

1.2.3 The Freezing Robot Problem

The freezing robot problem, first introduced in Trautman and Krause [1], highlights the deficiencies of existing predictive models that force the robot to come to a complete stop (or freeze). Trautman and Krause [1] shows that even under *perfect individual prediction* for all agents in the environment, the freezing robot problem can occur for certain configurations. Figure 1.1 shows a common scenario in human crowds, where people walking shoulder-to-shoulder towards the robot, forces it to go around the crowd or in the worst case, stop completely. As shown in Trautman and Krause [1], this happens as most existing navigation approaches ignore implicit cooperation between humans and the robot.

The key insight, which we will follow in this thesis, is that agents engage in *joint collision avoidance*, i.e., they adapt their trajectories to make room for other agents to navigate. The joint collision avoidance criteria has been shown to improve tracking of humans in dense crowds [1, 9, 10, 11].

The approach suggested in Trautman and Krause [1] to solve the freezing robot problem is to move away from individual agent prediction and model the joint distribution of

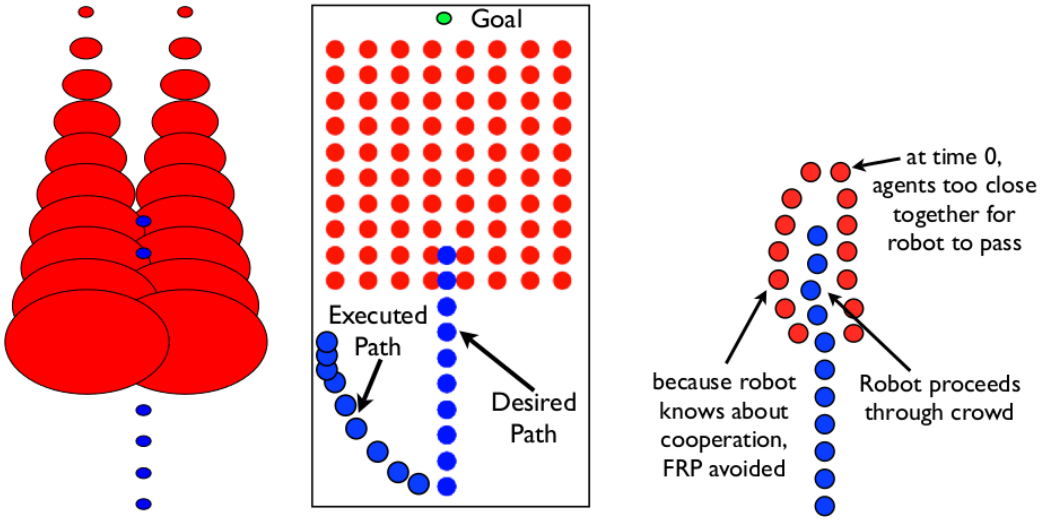


Figure 1.1: Example of Freezing Robot Problem (obtained from Trautman and Krause [1]). (left) With growing uncertainty about future trajectories of other agents, the robot cannot find any feasible path and *freezes*. (middle) Even in the case of perfect prediction, the robot executes a path that is highly suboptimal or sometimes infeasible. (right) When human-robot cooperation is accounted, the robot executes an optimal path through the crowd avoiding the Freezing Robot problem.

trajectories, as shown in equation 1.2. In addition to joint modeling, we need to model the robot as one of the agent so that we also include interactions between the robot and other agents. In other words, the prediction algorithm should model

$$P(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \cdots, \mathbf{f}^{(N)} | \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(N)}), \tag{1.3}$$

where $\mathbf{f}^{(R)}$ is the robot's trajectory. Note that this distribution encodes the idea of *cooperative planning* as it captures interactions among agents and the robot.

An interesting outcome from this modeling is that planning the robot's trajectory reduces to inference in this joint model, i.e. inferring what the robot should do given the actions of other agents

$$(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \cdots, \mathbf{f}^{(N)})^* = \underset{(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \cdots, \mathbf{f}^{(N)})}{\arg\max} P(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \cdots, \mathbf{f}^{(N)} | \mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}). \tag{1.4}$$

As we will see in Chapter 4, this joint model results in "human-like" behavior for the robot when modeling crowds, as we model the robot as one of the humans.

1.3 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 presents a brief survey of past works in the domain of robot navigation in human crowds. We also review existing works in the domain of human tracking in crowds and video surveillance. Chapter 3 presents our

novel approach for solving the problem of efficient path planning in dynamic environments when an accurate model of the world dynamics is known. We propose a heuristic-based graph search algorithm that results in safe and feasible paths for the robot in short planning times. Additionally, we present theoretical guarantees on the optimality of the resulting path and completeness of the planning algorithm. The task of obtaining an accurate model of the world dynamics (specifically, crowd dynamics) is tackled in Chapter 4, where we propose a novel statistical modeling approach that couples predictions for multiple agents through occupancy grids. The proposed model is learned from real world trajectory and can be used to predict future trajectories of dynamic agents in an environment. Finally, Chapter 5 summarizes the contributions of this thesis and provides directions for future research in this area.

Chapter 2

Navigation in Human Crowds : A Survey

In this chapter, we present a brief survey of past literature in the domain of robot navigation in human crowds. To address this problem, previous works either assume or construct a model of human motion, which is used to predict future trajectories. Given these predictions, they then proceed to plan the path for the robot to navigate through the crowd to its destination. Thus, the efficacy of these approaches depend on the accuracy of their human future trajectory predictions and robot path planning algorithms. As a part of this survey, we will present these aspects of the approaches in the context of our problem.

There has been a diverse set of works over the past two decades that have tackled the problem of robot navigation in human crowds. These approaches make varied assumptions, have different objectives and exhibit a wide range of results. We try to broadly classify these approaches according to their methodology, and highlight the benefits and drawbacks. More specifically, we will categorize the approaches based on their modeling assumptions and their planning objective. For each category, a brief description of the employed methodology is discussed in addition to its advantages and disadvantages. The intent of this chapter is to present the landscape of past research in this field to give some perspective and context to our proposed work presented in the coming chapters.

2.1 Taxonomy of Approaches

We broadly classify past works on the basis of their (1) objective of the planning algorithm, and (2) human motion model to predict future trajectories. The resulting classes are described in the following subsections.

2.1.1 Planning Objective

Planning the robot's path through the crowd involves several constraints that need to be satisfied. Collision-avoidance, dynamic feasibility and social compliance are some such constraints that typical path planning algorithms consider. Collision-avoidance is self-explanatory in that it requires the robot to avoid collisions with any obstacles including other agents in the crowd. Dynamic feasibility implies that the planned path needs to be executed by the

robot. In contrast, social compliance is a complex constraint that is hard to rigidly define. In broad terms, it implies that the resulting path for the robot needs to adhere to social norms followed by humans, thus making the path interpretable and predictable for humans in the crowd. Given these constraints, we can broadly classify past works as follows:

Safe Robot Navigation

The objective of this set of works involve the task of navigating a robot safely through a human crowds avoiding collisions and planning a dynamically-feasible path. Safety, in this context, refers to the collision-free aspect of the resulting path with respect to static and dynamic obstacles in the environment. As a result, these works do not consider the social aspects of navigation and hence, the resulting path of the robot is safe but may not be "human-like".

Social Robot Navigation

These works tackle the more difficult objective of not only safe robot navigation (as above), but also to move in a socially compliant way. Thus, the resulting robot paths are more predictable for the surrounding humans in the crowd.

Trajectory prediction

The set of works in this class do not necessarily involve robot navigation, but rather tackle the problem of accurately modeling human trajectories in crowds. As shown in Section 1.2.3, planning the path for the robot reduces to inference in this model, thereby obtaining a path for the robot that is "human-like" or socially-compliant. Most of the work in this category is from the domain of video surveillance tracking and computer vision.

2.1.2 Human Motion Model

For robots to navigate in human crowds, they need to employ a model of human motion in crowds so that accurate predictions of their future trajectories can be made. These predictions are then fed into a planning algorithm to plan the final trajectory for the robot to follow to navigate through the crowd. We can broadly classify past work based on the human motion model employed into four categories, that are discussed in subsequent subsections.

Independent Handcrafted Model (IH)

These approaches model each agent (or human) in the crowd independently of each other, i.e., they assume that the predictions for human trajectories are mutually independent. In addition to this assumption, the motion model is handcrafted (similar to a rule-based model) to match social behavior usually observed in crowds.

Independent Trained Model (IT)

Similar to the IH category, works in this class make the independence assumption but the model, instead of being handcrafted, is learned by training it on real-world human trajectory data.

Joint Handcrafted Model (JH)

Unlike the independent models, these works assume that the predictions are dependent on each other and jointly predict the trajectories of all interacting humans in the crowd. Most of these approaches don't model the joint distribution of trajectories explicitly, instead use some approximate handcrafted potential terms to capture the interactions.

Joint Trained Model (JT)

Similar to the JH class of works, these approaches jointly predict the trajectories of all humans in the crowd but the joint distribution is learned from real-world human trajectory data, and the learned model is used at inference time to make predictions.

In each of the categories in the above taxonomy, there are several related works. For conciseness purposes, we describe only the latest works that have been shown to perform better than others in their respective category. We would like to point out that our list of works is not exhaustive and doesn't list all the related past approaches. The taxonomy and the related works have been summarized in Table 2.1.

	IH	**IT**	**JH**	**JT**
Safe robot navigation	[12]	[13]	[11]	[14]
Social robot navigation	[15]	[16]	[17]	[18]
Trajectory prediction	-	[19, 20]	[21]	[22, 23]

Table 2.1: Taxonomy of related works, Independent (I), Handcrafted (H), Joint (J) and Trained (T)

2.2 Safe Robot Navigation

As described in the above section, the objective of these works is to navigate a robot through human crowds avoiding collisions and satisfying dynamic feasibility. Early works have dealt with this problem by using traditional handcrafted human motion models to obtain predictions for future trajectories. Most of these models make the simplistic assumption that motion of agents are independent of each other, except in close quarters where handcrafted potential terms predict collision-avoidance.

Hoeller et al. [12] employs a laser-based tracker to track humans in the crowd and combines the estimates with a potential field-based model to predict their future motion. These models have handcrafted quadratic repulsive potential terms that result in predictions that avoid obstacles and linear attractive potential terms that steer the predictions towards destination. These potentials are a function of the distances to other humans, obstacles and destination. Given these predictions of future trajectories, the path of the robot is planned using a variant of Expansive space trees (EST), Hsu et al. [24].

The potential field-based model captures simplistic human-space interactions such as obstacle avoidance, and works well in wide open spaces. The linear attractive potential terms capture the intent of the human to go towards the destination, but require the knowledge of the true destination of every human in the crowd. Although they are capable of

modeling obstacle avoidance, they fail at accurately capturing complex human-human interactions like cooperation. Another major drawback of such approaches is due to the independence assumption that doesn't account for the effect of robot's actions on the human trajectories.

On the other hand, Aoude et al. [13] uses similar independent human motion models but learned from real pedestrian data, rather than using handcrafted potential terms. The learned models are used to forecast future trajectories for humans in the crowd. More specifically, Aoude et al. [13] use Gaussian Processes (GP) to learn independent models of motion patterns of humans in real crowds. The future trajectories are grown using a variant of RRT to follow the motion pattern. Chance-constrained RRT, Luders et al. [25], is used to plan the robot's path guaranteeing probabilistic robustness to predicted human paths. The GP model is trained on human trajectories from real annotated crowd data.

Learning motion patterns from data enables Aoude et al. [13] to result in predictions that capture several navigation behaviors. By combining GPs and RRTs, it has a run-time that is suitable for real-time operation. Chance-constrained RRT, used in planning the path of the robot, guarantees probabilistically safe trajectories for the robot in the presence of humans. But the motion patterns modeled account for each individual human individually and do not capture human-human interactions. In dense crowds, where such interactions play a major role such a model would result in inaccurate predictions.

More recently, there have been several works which move away from the independence assumption and model the future trajectories of all interacting agents as a joint distribution. As a result, these works can capture human-human interactions within the crowd. Trautman et al. [11] introduced *Interacting Gaussian Processes* (IGP), which uses GPs to model individual trajectories of the pedestrians (and the robot) and couples their predictions using a handcrafted interaction potential term that aims to capture joint behaviors, such as cooperative collision avoidance. The robot's path is chosen as the MAP assignment to the modeled joint distribution (similar to Section 1.2.3).

The handcrafted interaction potential term explicitly assigns low probability mass to predictions that result in human-human and human-robot collisions, thus capturing joint collision avoidance. One important contribution of this work is that since the robot is part of the joint model, it also accounts for human-robot cooperation which was lacking in previous works. A major drawback of such an approach is the hand-tuned potential term that doesn't generalize across different environments and crowd settings.

To account for this drawback, one needs to learn the joint distribution from real-world crowd data without using hand-tuned potential terms. Kim et al. [14] introduced an online motion prediction method that learns per-agent motion models as the robot moves, with no prior knowledge of the environment. The prediction algorithm extends the reciprocal velocity obstacles approach, Van den Berg et al. [26], which captures joint collision avoidance among pedestrians. Given the predictions, the robot plans its own path using generalized velocity obstacles method, Wilkie et al. [27].

By learning individual motion model for each observed pedestrian, the online motion-prediction model can perform better than less responsive offline motion models. But the reciprocal velocity obstacles approach can only capture collision avoidance behavior and not cooperative behavior that is commonly observed in crowds. Also, the predictions are accurate only for short-term horizons and worsen over longer horizons.

2.3 Social Robot Navigation

Works in this category tackle the problem of coming up with socially-compliant paths in addition to safe and dynamic-feasibility. Social compliance implies that the resulting paths are predictable for surrounding humans in the crowd, which can be a difficult to define as an objective. Early works in this domain use rule-based systems which try to capture social norms that are typically observed in crowds.

Kirby et al. [15] presents an approach that tracks surrounding humans and uses the estimate of their current velocity to predict future locations. Pre-defined social conventions such as person avoidance, personal space, pass on the right, keeping a constant velocity etc. are encoded as social constraints on the robot's path. Given these constraints, A* is used to plan the robot's path. Since social norms such as passing on right and respecting personal space are explicitly encoded into the planning, the resulting path has social compliance respecting such behaviors. On the other hand, the constant velocity assumption used in human motion prediction doesn't hold true in real crowds and the pre-defined social conventions are specific to office hallways, not for general settings.

Rather than using these handcrafted rules to define social conventions, Kim and Pineau [16] learns social behaviors from human demonstrated crowd navigation behavior. They employ Bayesian inverse reinforcement learning (IRL), Ramachandran and Amir [28], to learn a cost function from human demonstrations obtained by tele-operating the robot in a real crowd. The features used to characterize this cost function are pedestrian speed, direction of his motion, local crowd density and distance to goal, and these are extracted from raw RGB-D sensor data. A low-level local path planner is used to optimize the cost function and plan an optimal path for the robot.

As the cost function is learned from human demonstrations, the robot learns to navigate in a socially compliant way and is generalizable to new environments. A reactive planner is used to account for any uncertainty regarding the pedestrian motion, which replans every time new sensor data is obtained. The major drawback of this approach is that future motions of the humans are independent of each other and hence, cannot capture human-human and human-robot interactions.

As argued before, to capture these interactions we need to model the joint distribution of the trajectories. Shiomi et al. [17] approximates this distribution by using a variant of the social forces model, Helbing and Molnar [29], which describes the interactions in terms of forces that correspond to objectives. Attractive forces guide the pedestrians towards their goal whereas repulsive forces ensure that collisions are avoided. These forces are a function of positions and speeds of the pedestrians. Given the predictions, the robot's path is planned using a local reactive planner.

In sparse crowds, approaches such as Shiomi et al. [17] that employ the social forces model have been shown to result in socially compliant paths. But in dense crowds, social force models fail as they cannot capture complex crowd behavior such as cooperation. This restricts the applicability of such approaches in dense crowds.

Kretzschmar et al. [18] learns parameters of a joint distribution model over all interacting agents using Maximum Entropy IRL, Ziebart et al. [30], from human demonstrations. The features used include acceleration, velocity, clearance, collision-avoidance, passing left vs right, group behavior etc. During prediction, they explicitly account for interactions between humans, and predict the future paths of both the humans and the robot jointly. Since the cost function is learned from real crowd data and the future paths are jointly predicted, this approach can capture interactions that are commonly seen in dense crowds resulting

in socially compliant path for the robot. Interestingly, the model also learns cooperative behavior between the humans and the robot. The major drawback is that the dimensionality of the feature vector scales with the number of agents in the crowd, making the approach scale poorly with the size of the crowd.

2.4 Trajectory Prediction

In this section, we will discuss works in the domain of video surveillance tracking and human trajectory prediction in crowds. These works are relevant as predicting trajectories of surrounding humans accurately is highly important to the task of navigation through crowds. Given such an accurate prediction model, we can plan the path of the robot using the same model to obtain socially-compliant paths that are "human-like".

Some of the works such as Joseph et al. [19] learn independent human motion prediction models by modeling motion patterns of pedestrians in real crowds. These motion patterns are modeled using GPs to regress over their (x, y) positions and a Dirichlet process (DP) to account for the unknown number of motion patterns. Activity forecasting introduced in Kitani et al. [20], on the other hand, infers traversable regions in a scene by modeling human-space interactions using semantic scene information. The traversability is defined through a cost function that is learned using Maximum Entropy IRL, Ziebart et al. [30], over the static semantic environment map from human trajectories.

Approaches that model motion patterns of pedestrians in environments capture human navigation behaviors and implicitly model environmental constraints on human motion (such as a static obstacle in the environment, that pedestrians avoid). Similarly, approaches like Kitani et al. [20] that explicitly model human-space interactions can learn more refined constraints on the motion according to the semantic objects in the environment. Unfortunately, both these sets of approaches don't account for human-human interactions i.e. they model each human independently of each other. Hence, they cannot capture cooperation or high-level social behavior among humans in the crowd.

As we already know, to account for human-human interactions we need to jointly predict the trajectories of all interacting agents in the crowd. Luber et al. [21] presented a pedestrian dynamics model based on Social Forces, Helbing and Molnar [29], that integrates the social forces model with a Kalman filter based multi-hypothesis tracker. The resulting model accounts for both inter-person influences and influences from static obstacles (using an occupancy map) in the environment. Thus, the trajectory predictions are accurate for humans in real crowds. One of the several drawbacks of such an approach, as discussed in the previous section, is that the use of social forces model has been shown to be effective only in sparse crowds and not in dense crowds. This makes it not readily applicable for any crowd scenario. Another important drawback of the model is that, as it doesn't infer the destination of the pedestrian, its long term prediction accuracy is low (Note that attractive forces that guide the pedestrian to the destination, which are a part of social forces, aren't used in this approach).

More recently, there have been approaches that learn a joint distribution over future trajectories of all interacting agents from real crowd data, rather than using handcrafted potential terms or forces. Pellegrini et al. [22] introduces a third order conditional random field (CRF) based approach to model the joint distribution of trajectories. The CRF-based approach is also able to identify groups within crowds on the basis of past trajectories. The parameters of the CRF are learned by training the model on annotated crowd datasets with

very dense crowds. Taking a similar approach, Alahi et al. [23] presents an LSTM-based approach that learns a deep recurrent generative model of the joint distribution accounting for interactions and short-term intentions. Given the past trajectories of all agents, this model couples their predictions through a *social pooling* layer that combines the hidden states of neighboring pedestrians.

These joint data-driven approaches perform well in real world dense crowd trajectory prediction tasks and capture important social aspects such as cooperation, joint collision avoidance and group memberships. Two important drawbacks of this approach are, (1) the human-space interactions, such as static obstacle avoidance, are not included in the model, and (2) they work only for short horizons as they are aimed to model interactions rather than achieve accurate long term predictions.

2.5 Summary

In this chapter, a brief survey of past literature in the domain of navigation and trajectory prediction in human crowds is presented. Approaches ranging from independent modeling with rule-based models to joint modeling using deep recurrent networks have been discussed along with their applicability and shortcomings. The intent of this chapter is to get a better understanding of the work that has been done and how that leads up to the work presented as a part of this thesis.

In the next chapter, Chapter 3, we present a novel planning algorithm that enables robots to navigate crowded dynamic environments, where the path of the dynamic obstacle is known. Such an algorithm can be used on a robot to re-plan its path upon receiving new sensory information and subsequent dynamic obstacle trajectory predictions. The problem of predicting the trajectory (especially, that of humans in crowds) is tackled in Chapter 4, where we present a statistical model that, given the past trajectories, predicts future trajectories accurately for long time horizons.

Chapter 3

Path Planning in Dynamic Environments with Adaptive Dimensionality

In this chapter, we present a novel path planning algorithm in densely populated dynamic environments where the trajectories of the dynamic obstacles are known. We apply the idea of adaptive dimensionality to speed up path planning in dynamic environments for a robot with no assumptions on its dynamic model. Specifically, our approach considers the time dimension in the search space only in those regions of the environment, where a potential collision may occur and plans in a low-dimensional state-space elsewhere. We show that our approach is complete and is guaranteed to find a solution, if one exists, within a cost sub-optimality bound. We experimentally validate our method on the problem of 3D vehicle navigation (x, y, heading) in dynamic environments. Our results show that the presented approach achieves substantial speedups in planning time over 4D heuristic-based A*, especially when the resulting plan deviates significantly from the one suggested by the heuristic. This chapter is adapted from our paper Vemula et al. [31] presented at SoCS 2016.

3.1 Introduction

It is important for mobile robots to be able to generate collision-free paths in environments that contain both static and dynamic obstacles. In static environments, robots can efficiently generate a collision-free path using the occupancy gridmap of the environment. But in dynamic environments, to account for the dynamic nature of obstacles, the robot needs to predict the future trajectories of these obstacles to plan its own path accordingly. These predictions involve a high degree of uncertainty due to sensor limitations and incorrect dynamic models. As a result, the predicted trajectories are subject to frequent changes due to incorrect predictions, which makes it necessary to generate new plans in a timely manner.

To account for dynamic obstacles in an environment, we need to include the time dimension into consideration. For example, planning a path for a non-holonomic robot in a dynamic environment involves a 4D state-space, i.e., (x, y, heading, time). Due to the curse of dimensionality, adding the time dimension substantially increases the number of states

to be searched, e.g., from 3D state-space considering only (x, y, heading), leading to long planning times especially, since there are potentially an unbounded number of timesteps for each spatial location.

The Adaptive Dimensionality (AD) approach, Gochev et al. [32], exploits the observation that while planning in a high dimensional space is needed to satisfy kinematic constraints and collision-free criteria, large portions of the path are still low dimensional. For instance, in planning for a non-holonomic robot, an optimal path generally includes straight-line segments that do not involve any turns or collisions with dynamic obstacles. This observation implies that high dimensional path planning is required only in the sections of the path where turning is required or where there is a potential collision with a dynamic obstacle.

Following this insight, we consider the time dimension only in those regions where a potential collision could occur and ignore it elsewhere. In this paper, we develop an approach that can achieve speedups over full-dimensional heuristic-based A* without any assumptions on robot capabilities by employing a variant of the Adaptive Dimensionality approach.

Given a path planning problem in a high dimensional space, it is possible to find an optimal solution through a complete search. For example, heuristic-based A* variant algorithms exist that are guaranteed to find an optimal solution, [33]. Because these algorithms rely on low-dimensional heuristics, search can be counter-intuitive.

Consider the example shown in Figure 3.1, where the resulting path (dash-dot path), in the absence of the dynamic obstacle (disc), is towards the heuristic. But, in the presence of the dynamic obstacle, this path is in collision and cannot be executed. Hence, we need to come up with the alternative path (dashed path), which is against the heuristic. Heuristic-based A* would expand a large number of states and will take a long time to generate the new path whereas our approach generates the alternative path quickly, because it plans in a lower dimensional space.

More generally, we observe that substantial sections of paths found are not in collision with any dynamic obstacles, implying that we need not consider the time dimension in such regions. We can obtain quicker planning times by planning in low dimensional state-space for those regions and in full dimensional state-space only where it is necessary to reason about a potential collision with an obstacle.

Based on these observations, we explore the idea of adaptive dimensionality to solve the target problem.

In the remainder of this chapter, we will give an overview of relevant existing work in Section 3.2. The planning problem is formally defined in Section 3.3. Section 3.4 will describe our approach and prove the theoretical guarantees. The efficiency of the method is demonstrated by applying it to a 3D non-holonomic robot navigation problem in the presence of dynamic obstacles, showing a significant increase in speed over 4D heuristic-based A* planner for this task, in Section 3.5.

3.2 Related Work

Our work is relevant to path planning in dynamic environments and works on coping with high dimensionality. In general, we divide the existing approaches into three categories: work that deals with planning in dynamic environments, work that deals with high-dimensional planning using adaptive dimensionality and work that uses hybrid di-

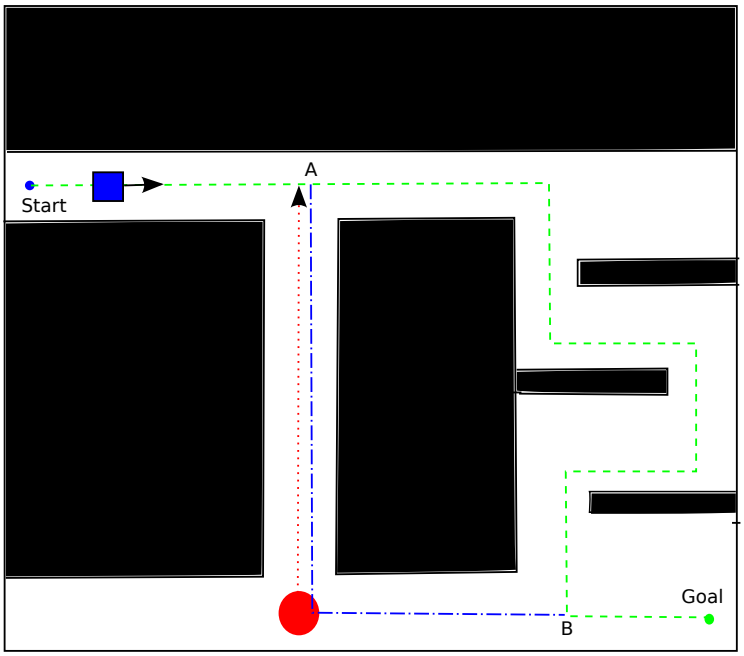


Figure 3.1: Example of a dynamic environment where the heuristic leads the robot (blue square) into collision (on blue dash-dot path) with the dynamic obstacle (red disc). We need to find an alternate path (green dashed path) from A to B without expanding a large number of states.

mensionality path planning in dynamic environments.

3.2.1 Path Planning in Dynamic Environments

A common approach used for efficient path planning in dynamic environments involves modeling moving obstacles as static objects with a small window of high cost around the beginning of their projected trajectories [34, 35]. By avoiding the additional time dimension, these approaches can efficiently find paths that do not collide with any obstacles in the near future. However, they can suffer from severe sub-optimality or even incompleteness due to the uncertainty of moving obstacles in the future.

To plan and re-plan online, several approaches have been suggested that sacrifice near-optimality guarantees for efficiency [36], including sampling-based planners such as RRT-variants that can quickly obtain kinodynamically feasible paths in a high dimensional space [37, 38]. However, these sampling-based approaches do not provide any global optimality guarantees that we require in most cases.

Other approaches delegate the dynamic obstacle avoidance problem to a local reactive planner which can effectively avoid collision with dynamic obstacles [2, 3]. These methods have the disadvantage that they can get stuck in local minima and are generally not globally optimal.

Among the works that provide global optimality guarantees that are relevant to the presented work, HCA* [33] is an approach that plans in the full space-time search space

for a path from start to goal, taking dynamic obstacles into account under the guidance of a low-dimensional heuristic. In dynamic environments, HCA* provides guarantees on optimality and can be applied to path planning for a robot without any assumptions on its motion model.

Recently, approaches such as SIPP and its variants [39, 40], have been introduced that obtain fast, globally optimal paths in dynamic environments. But, SIPP assumes that the robot is capable of waiting in place. In cases where this assumption doesn't hold, SIPP is essentially a full space-time A* planner. Thus, the advantages from this algorithm are restricted to only those robots which have the capability of waiting in place, unlike a fixed wing aircraft or a motorcycle. Besides, when fuel efficiency is included in the cost, fuel consumption is generally higher during idling than moving.

We use HCA* as our baseline algorithm as it doesn't make any assumptions on the motion model of the robot and provides optimality guarantees, similar to our approach.

3.2.2 Adaptive Dimensionality

To accelerate planning, a variety of algorithms try to avoid global planning in high-dimensional state-space. In these algorithms, planning is split into a two-layer process where a global planner deals with a low-dimensional state-space and provides an input to a high-dimensional local planner [41]. The local planner is a reactive planner that avoids obstacles locally and hence, is fast and efficient. However, these approaches can result in paths that are highly suboptimal or that cannot be executed, due to mismatches in the assumptions made by the global and local planners.

In Knepper and Kelly [42], highly accurate heuristic values are computed by solving a low-dimensional problem and are then used to direct high-dimensional planning. However, this approach does not explicitly decrease the dimensionality of the state-space and can lead to long planning times when the heuristic is incorrect.

By contrast, the Adaptive Dimensionality (AD) approach, Gochev et al. [32], explicitly decreases the dimensionality of the state-space in regions where full-dimensional planning is not needed. This approach introduces a strategy for adapting the dimensionality of the search space to guarantee a solution that is still feasible with respect to a high dimensional motion model while making fast progress in regions that exhibit only low-dimensional structure. In Gochev et al. [43], path planning with adaptive dimensionality has been shown to be efficient for high-dimensional planning such as mobile manipulation. The AD approach has been extended in Gochev et al. [44], to get faster planning times by introducing an incremental planning algorithm. Zhang et al. [45] extends this method in the context of mobile robots by using adaptively dimensional state-space to combine the global and local path planning problem for navigation. Our approach builds on the AD approach and applies it to path planning in dynamic environments.

3.2.3 Hybrid Dimensionality in Dynamic Environments

Some approaches only plan in full-dimensional space-time search space until the end of an obstacle's trajectory and then finish the plan in a low-dimensional state-space. Time-bounded lattice planning, [46], neglects dynamic obstacles and the time dimension in the search space after a certain point in the time. Several works, [47, 48], have extended this algorithm to account for kinematic and dynamic feasibility in the resulting paths by using

a hybrid dimensionality state-space. These approaches sacrifice optimality for faster planning times and don't provide theoretical guarantees on the sub-optimality of the solution. In contrast, our algorithm doesn't prune the dynamic obstacle trajectories, takes the entire obstacle trajectories into account and returns a bounded sub-optimal collision-free path. By considering the entire trajectory of the obstacles the proposed algorithm ensures a globally optimal solution.

3.3 Problem Definition

In this chapter, we follow the simplifying assumptions used in Phillips and Likhachev [39] that the trajectories of moving obstacles are known and that obstacles move at a constant speed. Based on these assumptions, path planning in a dynamic environment can be formulated more generally as path planning in a high-dimensional space as follows. A path planning problem is defined as a tuple $\Phi = [G = (S, T), c, X_s, X_g]$, where G denotes a graph consisting of S, a set of discretized states in a d-dimensional space, and T, a set of feasible transitions between each pair of states $X_i, X_j \in S$; c, a function encoding a non-negative cost $c(X_i, X_j)$ for each pair of transitions $(X_i, X_j) \in T$; $X_s \in S$, a start state, and $X_g \in S$, a goal state. For instance, the target problem for a ground vehicle can be defined in 4-D state space (x, y, θ, t) where each variable denotes x-coordinate, y-coordinate, vehicle heading, and time, respectively. Note that transitions that result in collision with an obstacle are assigned infinite cost, making them invalid.

A path between states X_i and X_j is denoted by $\pi(X_i, X_j)$, and the cost of a path is defined as the sum of all transition costs in the path.

Given a planning problem $\Phi = [G, c, X_s, X_g]$, the goal is to find a minimum cost path between the two states X_s and X_g, denoted by $\pi^*(X_s, X_g)$. Alternatively, given a suboptimality bound ϵ, the goal of the planner can be relaxed to find a path $\pi(X_S, X_G)$ such that its cost $c(\pi(X_S, X_G)) \leq \epsilon \cdot c(\pi^*(X_S, X_G))$.

3.4 Approach

We describe the adaptive dimensionality approach used for path planning in dynamic environments, and the algorithm for finding a bounded cost sub-optimal path.

3.4.1 Adaptive Dimensionality for Dynamic Environments

Our approach follows the algorithm for planning with adaptive dimensionality introduced in Gochev et al. [32]. Following their notation, the target problem in Section 3.3 can be rewritten as follows. Graph G is substituted with the adaptive-dimensionality graph $G^{ad} = (S^{ad}, T^{ad})$. G^{ad} is constructed from two graphs: a high dimensional graph $G^{hd} = (S^{hd}, T^{hd})$ with dimensionality h and a low dimensional graph $G^{ld} = (S^{ld}, T^{ld})$ with dimensionality l. The state-space S^{ld} is a projection of S^{hd} onto a lower dimensional manifold (h>l) through a projection function:

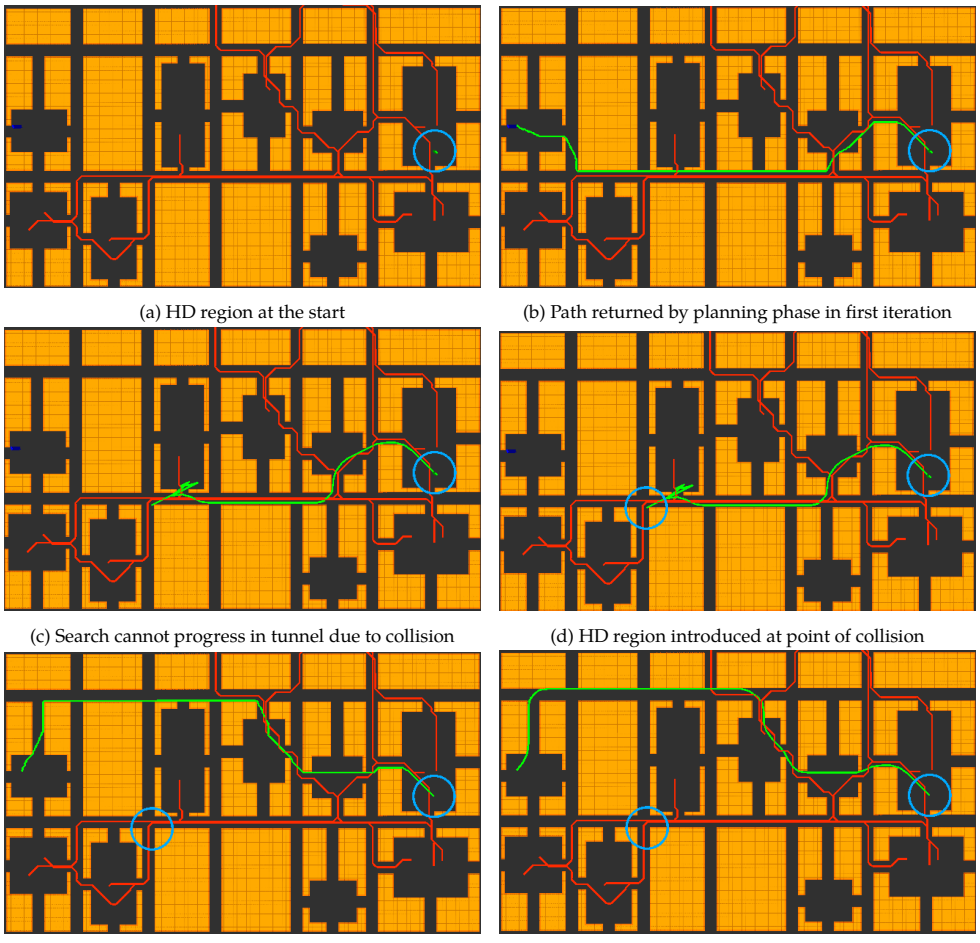
$$\lambda : S^{hd} \rightarrow S^{ld}. \tag{3.1}$$

Similarly, an inverse projection function $\lambda^{-1} : S^{ld} \rightarrow \mathcal{P}(S^{hd})$ is defined to map low-dimensional states to the set of all their high-dimensional pre-images,where $\mathcal{P}(S^{hd})$ denotes the power set of S^{hd}.

Both state spaces S^{hd} and S^{ld} can have their own transition sets T^{hd} and T^{ld}, with a constraint that transitions in a high-dimensional space are more expensive than the corresponding transitions in a low-dimensional space, that is for every pair of states X_i and X_j in S^{hd},

$$c(\pi^*_{G^{hd}}(X_i, X_j)) \geq c(\pi^*_{G^{ld}}(\lambda(X_i), \lambda(X_j))). \tag{3.2}$$

We note that this constraint is important for bounding the suboptimality that will be discussed later in Section 3.4.3.



(a) HD region at the start

(b) Path returned by planning phase in first iteration

(c) Search cannot progress in tunnel due to collision

(d) HD region introduced at point of collision

(e) Path returned by planning phase in second iteration

(f) Tracking successful and path returned as solution

Figure 3.2: Example run of the algorithm on a sample map. HD regions are indicated by blue circles, paths of dynamic obstacles by red lines and path found using our approach by green line.

In our target problem of planning in a dynamic environment, the low-dimensional state-space S^{ld} consists of only spatial state variables, e.g., xy-coordinates, and the high-dimensional state-space S^{hd} consists of states with spatio-temporal variables including a time dimension. Theoretically, the time dimension is unbounded and thus, the high-dimensional graph G^{hd} is an infinite graph. For practical purposes, we bound the time dimension by a upper bound

T, which would slightly modify the goal of planning problem into finding a least-cost path that can reach the goal from start within time T. For any high dimensional state $X^{hd} \in S^{hd}$, we will use the notation $t(X^{hd})$ to denote the value of time dimension associated with that state.

The projection function λ projects the high-dimensional state X^{hd} to a low-dimensional state X^{ld} with only the spatial variables. If we follow the original definition of λ in Equation 3.4.1 then, for a given low-dimensional state X^{ld}, the inverse projection function λ^{-1} would map state X^{ld} to the set of all X^{hd} where the spatial configuration of X^{hd} is the same as X^{ld} and $0 \leq t(X^{hd}) \leq T$. Thus, for each low-dimensional state, there are T corresponding high-dimensional states, which is quite a large number as T is usually a high value.

Here, we introduce a pruning technique based on an observation that not all of high-dimensional states are reachable from the start state. For example, consider a low-dimensional state $X \in S^{ld}$ which is mapped to T high-dimensional states. If the time-optimal path from start to this state, ignoring dynamic obstacles, reaches at time t_f then all the states X^{hd} with $\lambda(X^{hd}) = X$ and $t(X^{hd}) < t_f$ are essentially unreachable and hence, can be pruned away from the search space.

Taking advantage of this fact, we decrease the size of search space by performing a low-dimensional time-optimal Dijkstra search in G^{ld}, which ignores dynamic obstacles, initially from the start state to all low-dimensional states and keep track of the time at which we reach each state. We store this time as a dependent variable t_{dep} of the low-dimensional state and ignore all the corresponding high-dimensional states whose time value is less than t_{dep} in the inverse projection mapping. Note that this dependent time variable need not be the exact optimal time obtained from the Dijkstra search, it just needs to be a lower bound on the optimal time. Pruning the search space in this way is necessary as it speeds up the planning by a considerable amount while still maintaining the completeness property of the planning algorithm.

Thus, we define the inverse projection function, λ^{-1} as:

$$\lambda^{-1}(X^{ld}) = \{X^{hd} \mid \lambda(X^{hd}) = X^{ld}, t_{dep} \leq t(X^{hd}) \leq T\}$$

, where t_{dep} is the dependent time variable associated with the low-dimensional state X^{ld}.

The low-dimensional transition set is $T^{ld} = \{(X_i, X_j) | X_i, X_j \in S^{ld}\}$ where it is feasible for the robot to move from the spatial configuration of X_i to X_j according to its motion model. The transition set $T^{hd} = \{(X_i, X_j) | X_i, X_j \in S^{hd}\}$ where, $t(X_j) \geq t(X_i)$ and it is feasible for the robot to move from spatial configuration of X_i to X_j in time $(t(X_j) - t(X_i))$ according to its motion model. Note that we can check for collisions with any dynamic obstacle only in the high-dimensional transitions as we have the time information.

3.4.2 Algorithm

The planning algorithm follows that of Gochev et al. [32]. Here, we sketch the general algorithm and describe how it has been applied to our target problem of handling dynamic obstacles.

Adaptive Dimensionality Graph Construction

The algorithm iteratively constructs S^{ad}, starting with S^{ld} and introducing high-dimensional regions in subsequent iterations. Once a high-dimensional region is introduced we replace

all the low-dimensional states that fall inside it with their high-dimensional counterparts as given by λ^{-1} to get the re-constructed S^{ad} for the next iteration. The transition set T^{ad} is also iteratively constructed, starting with T^{ld} and re-constructed as follows in subsequent iterations. For any state $X_i \in S^{ad}$:

- If X_i is high-dimensional then, for all high-dimensional transitions $(X_i, X_j^{hd}) \in T^{hd}$, if $X_j^{hd} \in S^{ad}$ then $(X_i, X_j^{hd}) \in T^{ad}$. Otherwise, $(X_i, \lambda(X_j^{hd})) \in T^{ad}$.

- If X_i is low-dimensional then, for all low-dimensional transitions $(X_i, X_j^{ld}) \in T^{ld}$, if $X_j^{ld} \in S^{ad}$ then $(X_i, X_j^{ld}) \in T^{ad}$, and for all high-dimensional transitions $(X, X_j^{hd}) \in T^{hd}$ where $X \in \lambda^{-1}(X_i)$, if $X_j^{hd} \in S^{ad}$ then $(X_i, X_j^{hd}) \in T^{ad}$.

Main Loop

We start with G^{ad} same as G^{ld} and a high-dimensional region added at the start, which is necessary as the start state X_S is high-dimensional with $t(X_S) = 0$. Note that the goal state X_G is not high-dimensional as we don't know the value of the time dimension for the goal state.

AD planning phase. At the start of each iteration, the current graph G^{ad} is searched for a path $\pi^*_{G^{ad}}$ from the start to the goal, using a suboptimal graph search algorithm like weighted A* with a suboptimality bound ϵ_{plan}. During the search for this path, we consider the dynamic obstacles only in high-dimensional regions of S^{ad} and not in the low-dimensional regions. Hence, the path found could potentially be in collision with a dynamic obstacle in the low-dimensional regions. If no path $\pi^*_{G^{ad}}$ is found, we return that there exists no feasible path that can reach the goal from start within time T, and the algorithm terminates.

Tracking phase. If a path is found, then in the tracking phase, a high-dimensional tunnel τ is constructed around the path $\pi^*_{G^{ad}}$ and searched for the least-cost path $\pi^*_\tau(X_S, X_G^{hd})$ where $X_G^{hd} \in \lambda^{-1}(X_G)$. The tunnel is constructed by projecting all the states within to their high-dimensional counterparts. Notice that since the tunnel is entirely high-dimensional and is a subgraph of G^{hd}, we consider dynamic obstacles in the entire tunnel and hence, the path found is guaranteed to be feasible and collision-free. If a path π^*_τ is found and its cost is less than $\epsilon_{track} * c(\pi^*_{G^{ad}})$, then it is returned as the solution by the algorithm and the algorithm terminates. If no path is found, then we identify the farthest location in the tunnel until which the planner has progressed (i.e. the path with most progress), introduce a high-dimensional region there and move onto next iteration. If a path is found and its cost is greater than $\epsilon_{track} * c(\pi^*_{G^{ad}})$, then we identify the location where the largest cost discrepancy (cost difference) between the path π^*_τ and $\pi^*_{G^{ad}}$ is observed and a high-dimensional region is introduced there. In both cases, if we identify a location which is already high-dimensional, then the size of the high-dimensional region at that location is increased.

Graph updating phase. The algorithm re-constructs G^{ad} based on the new high-dimensional regions introduced and moves onto the next iteration of planning and tracking, and keeps repeating until it finds a feasible, collision-free path or returns that there is no such path. An example run of the algorithm is shown in Figure 3.2. The algorithm is presented in Algorithm 1.

Algorithm 1 Planning with AD in dynamic environments

1: $G^{ad} = G^{ld}$
2: AddFullDimRegion(G^{ad}, $\lambda(X_S)$)
3: **loop**
4: Search G^{ad} for the path $\pi^*_{G^{ad}}(X_S, X_G)$
5: **if** no $\pi^*_{G^{ad}}(X_S, X_G)$ is found **then**
6: **return** no path from X_S to X_G within time T exists
7: **end if**
8: Construct tunnel τ around $\pi^*_{G^{ad}}(X_S, X_G)$
9: Search τ for the least-cost path $\pi^*_\tau(X_S, X^{hd}_G)$ where $X^{hd}_G \in \lambda^{-1}(X_G)$
10: **if** no $\pi^*_\tau(X_S, X^{hd}_G)$ is found **then**
11: Let $\pi(X_S, X_{end})$ be the path with most progress
12: **if** X_{end} is high-dimensional **then**
13: GrowFullDimRegion(G^{ad}, $\lambda(X_{end})$)
14: **else**
15: AddFullDimRegion(G^{ad}, X_{end})
16: **end if**
17: **else if** $c(\pi^*_\tau(X_S, X^{hd}_G)) > \epsilon_{track} * c(\pi^*_{G^{ad}}(X_S, X_G))$ **then**
18: Identify state X_r with largest cost discrepancy
19: **if** X_r is already high-dimensional **then**
20: GrowFullDimRegion(G^{ad}, $\lambda(X_r)$)
21: **else**
22: AddFullDimRegion(G^{ad}, X_r)
23: **end if**
24: **else**
25: **return** $\pi^*_\tau(X_S, X^{hd}_G)$
26: **end if**
27: **end loop**

3.4.3 Theoretical Properties

The given algorithm is complete with respect to the underlying graph G^{hd} and provides a sub-optimality bound on the cost of the returned path.

Theorem 1 *If a path $\pi^*_\tau(X_S, X^{hd}_G)$ is found in the tracking phase, it is guaranteed to be collision-free with respect to all obstacles.*

Proof The tunnel τ constructed around $\pi^*_{G^{ad}}$ is entirely high-dimensional and is a subgraph of G^{hd} therefore, the search space considers the transition set T^{hd}. In T^{hd}, transitions that are in collision with dynamic obstacles are assigned infinite cost, essentially making them invalid. Hence, the path found in the tunnel π^*_τ is guaranteed to be collision-free with respect to all obstacles □

Theorem 2 *If no path $\pi^*_{G^{ad}}(X_S, X_G)$ is found in the planning phase, then no collision-free, feasible path exists from start to goal in G^{hd} that can reach the goal from the start within time T.*

Proof If no path is found during the planning phase of the first iteration, no feasible path

exists in the absence of dynamic obstacles (Note that we only consider dynamic obstacles in the high-dimensional regions during the planning phase). Therefore, there is no collision-free path. If no path is found during the planning phase of any subsequent iteration, then the algorithm was not able to progress in a high-dimensional region. It cannot be a low-dimensional region because in such a case, it would have terminated in a previous iteration. If the algorithm is not able to progress in a high-dimensional region, then all the transitions into or inside the region are blocked by dynamic obstacles. Because we allow transitions to all X^{hd} inside the region where $t_{dep} \leq t(X^{hd}) \leq T$ and we already know that the planner cannot reach X^{hd} at a time earlier than t_{dep}, that guarantees that there exists no path in G^{hd} that can reach the goal from start within time T. □

Theorem 3 *The algorithm always terminates after a finite number of iterations.*

Proof If no path is found at the end of a iteration, we introduce either a new high-dimensional region or increase the size of an existing one. As the time dimension is bounded above by T, we have a finite state-space. Hence, in the worst scenario, after a finite number of iterations G^{ad} will be the same as G^{hd} and the algorithm will either terminate with a feasible path or return that there is no feasible, collision-free path. □

Theorem 4 *If a path $\pi(X_S, X_G^{hd})$ is found at the time of termination, its cost is no more than $\epsilon_{plan} * \epsilon_{track} * c(\pi^*_{G^{hd}}(X_S, X_G^{hd}))$ where $\pi^*_{G^{hd}}(X_S, X_G^{hd})$ is the optimal least-cost path in G^{hd}.*

Proof We obtain this bound using equation 3.2 and the proof is similar to that of the AD approach. For this proof, we refer the reader to Gochev et al. [32]. □

3.5 Evaluation

3.5.1 Experimental Setup

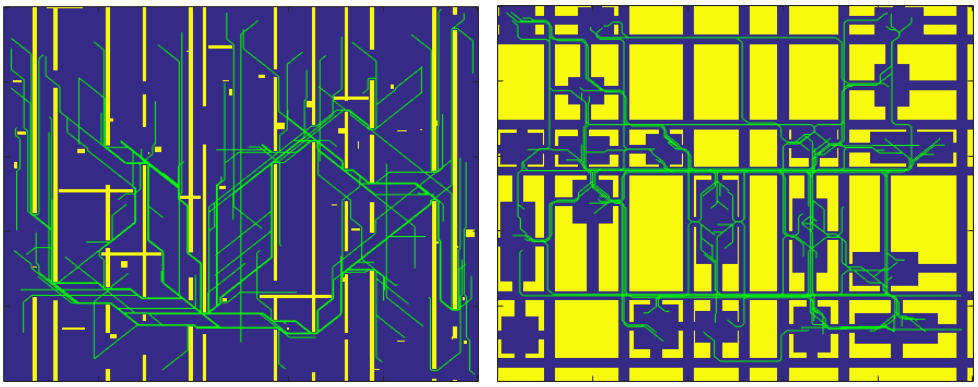
For an experimental evaluation of the presented approach, we use the domain of robotic path planning in dynamic environments for a 3D-(x, y, θ) non-holonomic robot. To successfully avoid dynamic obstacles in the environment, we will need to add the time dimension to the state-space while planning. Hence, the full-dimensional planning has a 4D (x, y, θ, t) state-space. Our implementation of the algorithm kept track of the high-dimensional regions in the environment as spheres: 2D (x, y) circles, in the 4D planning case, as this allowed to quickly check if a state falls inside a region or not, and also quickly add new regions or grow existing regions.

We modeled our environment as a planar world and the robot as a polygonal object with a unicycle motion model, which doesn't allow waiting in place actions. Our adaptive-dimensionality space consists of a 2D (x, y) low-dimensional state-space and a 4D (x, y, θ, t) high-dimensional state-space, where θ is the heading of the robot. Thus the projection function is:

$$\lambda(x, y, \theta, t) = (x, y).$$

We used a set of 16-discretized values for the heading angle and a maximum value of $T = 1000$ seconds at a resolution of 0.1 seconds for the time dimension. The set of motion primitives used for the 4D states consists of pre-computed kinematically feasible motion sequences as used in a lattice-type planner [34]. The motion primitives used for the 2D states

were the eight neighboring states according to the eight-connected 2D grid. Note that the motion primitives for 2D states do not produce feasible paths that can be executed by the robot. The objective of the planner is to find the minimal time path from the start state to goal state. Hence, cost of each edge in the graph is the time taken to execute the respective action multiplied by a constant factor.



(a) Maze-like environment (b) Indoor environment

Figure 3.3: Example maps, with paths (green) of dynamic obstacles shown, used in our experiments. Static obstacles are shown in yellow and free space in blue.

We compared our algorithm to the baseline 4D HCA* planner on several different environment sizes. In small environments with a few hundred cells along each spatial dimension, the baseline planner comes up with the plan quickly, so there is no advantage from our approach. In very large environments with more than 4000 cells along each spatial dimension, the baseline planner runs out of memory to find a solution, while our approach, since it deals with a low-dimensional state-space, was still able to plan successfully. To effectively compare the two approaches at the same level, we chose a moderate environment size of 2500 cells along each spatial dimension and generated 50 maze-like random environments like the one shown on the left in Figure 3.3. We also generated 50 random indoor environments of the same size like the one shown on the right in Figure 3.3. These indoor environments are composed of a series of narrow hallways and rooms on a grid placed randomly, while the maze-like environments are composed of a series of walls with small gaps in them to allow the robot to pass through.

In each of these environments, we randomly generate 30 dynamic obstacles. Each dynamic obstacle could come in a large or small size, randomly chosen, and started at a random configuration in the environment. To generate the trajectory for a dynamic obstacle, random goals were chosen and 2D A* is used to generate the paths between the goal points. We chose the start and goal configuration for each dynamic obstacle so that the resulting path is long enough, ensuring that they cover a significant area of the environment. In the indoor environments, the large dynamic obstacles fill the entire width of the hallways, so there is no way to pass them while the narrow dynamic obstacles fill half the width of the hallway, so they can be passed. In the maze-like environments, the dynamic obstacles traverse through the small gaps that the robot tries to pass through, resulting in congestion at such gaps. For each set of environments, we execute two sets of runs - one with 10 dynamic obstacles in each environment and the other with an additional 20 dynamic obstacles

Algorithm	Number of Success	Epsilon	time (secs)		# 4D expands		# 2D expands		path cost	
			mean	std dev	mean	std dev	mean	std dev	mean	std dev
Adaptive	49	1.1	6.4	0.7	3160	1105	10810	9361	39442	4438
4D	7	1.1	99.3	67.7	127393	87024	0	0	37142	5766
Adaptive	50	1.5	20.9	48.5	16688	42804	33276	88880	55342	14668
4D	40	1.5	67.1	75.8	85324	96306	0	0	49150	11568
Adaptive	50	2.0	18.4	39.2	16029	42418	23193	62865	60050	17148
4D	44	2.0	36.5	61.5	45172	76970	0	0	54090	15339

Table 3.1: Results on 50 indoor environments with 10 dynamic obstacles.

Algorithm	Number of Success	Epsilon	time (secs)		# 4D expands		# 2D expands		path cost	
			mean	std dev	mean	std dev	mean	std dev	mean	std dev
Adaptive	41	1.1	6.7	0.8	3705	1379	12524	9901	40740	2200
4D	5	1.1	91.0	71.2	111165	87228	0	0	38320	6522
Adaptive	40	1.5	11.7	14.0	7318	9285	21454	38559	54690	16811
4D	21	1.5	70.3	86.7	88576	109859	0	0	47566	9916
Adaptive	46	2.0	18.5	26.6	16000	31148	13672	21383	57760	19450
4D	23	2.0	35.8	69.8	43546	86677	0	0	50039	12256

Table 3.2: Results on 50 indoor environments with 30 dynamic obstacles.

(making it a total of 30 moving obstacles) in each environment.

The underlying search algorithm used in both the planning and tracking phase is weighted A* with the ϵ_{plan} sub-optimality bound. The tunnel width used for the tracking phase, was 10 cells, and the radii of the newly added spheres were 20 cells. For the heuristic used by the weighted A* planners, in our approach and the baseline HCA*, we use a 2D Dijkstra search from the goal state to all the (x, y) cells in the environment ignoring the dynamic obstacles. During the computation of heuristic, static obstacles are inflated by the inscribed circle radius of the robot to preclude paths through areas that are too narrow for the robot to physically traverse.

For each environment, we try three values of ϵ: 1.1, 1.5, and 2 with the adaptive planner using the square root of ϵ for both ϵ_{plan} and ϵ_{track}, thus giving an overall sub-optimality bound of ϵ for the adaptive planner. We use the same set of ϵ values for the baseline planner and compare their performance. For both planners, we enforce a maximum planning time of 5 minutes for all ϵ values. The code for our algorithm, the baseline algorithm and the experiments can be found at https://github.com/vvanirudh/sbpl_dynamic_adaptive_planner.

3.5.2 Results

For both sets of environments, we compare the planning time, number of high-dimensional (4D) states expanded, number of low-dimensional (2D) states expanded and resulting path cost, between our approach and the baseline 4D HCA* approach. We also list out the number of cases among the set of 50 environments that our approach could come up with a solution within 5 minutes of planning time and the number of cases the baseline approach could. Note that statistics like mean planning time, number of HD expansions, number of LD expansions and path cost, are computed only on cases where both approaches could come up with a solution within 5 minutes.

Tables 3.1 and 3.2 present the results for 50 randomly generated indoor environments with 10 and 30 dynamic obstacles respectively. In these environments, the low-dimensional

Algorithm	Number of Success	Epsilon	time (secs)		# 4D expands		# 2D expands		path cost	
			mean	std dev	mean	std dev	mean	std dev	mean	std dev
Adaptive	47	1.1	7.9	1.4	5105	2177	26665	9660	432425	43683
4D	4	1.1	161.5	59.8	195758	79955	0	0	416650	40272
Adaptive	48	1.5	14.2	11.2	9622	8566	22588	21539	532652	91234
4D	45	1.5	41.3	43.8	51515	56488	0	0	562109	95470
Adaptive	48	2.0	12.6	16.6	11771	13531	25630	34279	537873	89790
4D	48	2.0	18.6	35.8	22485	45048	0	0	622739	104136

Table 3.3: Results on 50 maze-like environments with 10 dynamic obstacles.

Algorithm	Number of Success	Epsilon	time (secs)		# 4D expands		# 2D expands		path cost	
			mean	std dev	mean	std dev	mean	std dev	mean	std dev
Adaptive	46	1.1	18.2	16.1	12565	12843	50012	12470	441100	91923
4D	2	1.1	192.7	159.0	236515	196442	0	0	424250	81529
Adaptive	48	1.5	25.4	33.8	16558	18703	29679	28739	524451	83024
4D	45	1.5	46.7	46.0	59116	59870	0	0	553686	89470
Adaptive	48	2.0	22.9	36.2	18922	22002	44550	107434	538370	92375
4D	48	2.0	21.3	34.7	25986	43153	0	0	623997	103201

Table 3.4: Results on 50 maze-like environments with 30 dynamic obstacles.

heuristic used is very often misleading as it cannot account for dynamic obstacles and leads the search into a blocked hallway. For $\epsilon = 1.1$, the planning problem is hard and the baseline could solve only 5 environments with 30 dynamic obstacles (7 in the case of 10 dynamic obstacles). In comparison, our approach could solve 41 of the 50 environments with 30 dynamic obstacles (49 in the case of 10 dynamic obstacles) with a substantially smaller mean planning time. As the ϵ value increases, the planning problem becomes easier and performance of the baseline approach improves. Even in these easier cases, our approach has a comparable performance, if not better than that of baseline. Results across tables 3.1 and 3.2 show that our approach performs well even when the number of obstacles increases, whereas the performance of baseline degrades substantially.

The results for the 50 randomly generated maze-like environments with 10 and 30 dynamic obstacles are presented in tables 3.3 and 3.4. These environments are characterized by tight turns and potential dynamic obstacle collisions at the gaps in the walls. In most cases, the robot would have to swerve around the obstacle to avoid collision. Hence, the resulting path doesn't deviate significantly from the one suggested by heuristic. From the results we can observe that when the planning problem is difficult (for example, when $\epsilon = 1.1$), our approach could solve a large number of cases (47 and 46 of 50) when compared to the baseline (4 and 2 of 50). But as the ϵ value increases and the planning problem becomes easier, performance of baseline quickly catches up with our approach and in one of the case, outperforms our approach as well ($\epsilon = 2$ in the 30 dynamic obstacles case). But in most runs, our approach performs better than the baseline in mean planning time and the path cost. Results across tables 3.3 and 3.4 show that there is not as much decrease in the performance of baseline with increase in number of obstacles, when compared to the indoor environments.

3.6 Discussion

Interestingly, in indoor environments our approach returns paths with higher costs (but still within the suboptimality bound) when compared to the baseline approach. This is due

to the fast low-dimensional 2D planning used in our approach which when a hallway is blocked finds an alternative path through an adjacent hallway, even if it is against heuristic. In contrast, the baseline tries to go through the blocked hallway suggested by the heuristic by wasting time before and entering the hallway once the obstacle comes out. Hence, the path returned by baseline often has low cost.

Generally, in environments where dynamic obstacles do not block the path suggested by heuristic, the baseline approach is fast and often outperforms our approach. This is the case in maze-like environments where the robot has to just swerve around the obstacle to avoid it. Hence, we see a good performance of baseline algorithm in such cases. But in cases where the solution required a path significantly different from the one computed by heuristic, baseline performs poorly and our approach outperforms it. This is the case in indoor environments where the entire hallway is blocked by an obstacle and the planner has to find an alternative path which might be against the heuristic. In such environments, as the number of dynamic obstacles increases, the heuristic becomes less informative and performance of baseline degrades. Our approach circumvents this through its iterative nature and fast low-dimensional planning.

3.7 Summary

In this chapter, we have presented a new approach to path planning in dynamic environments that doesn't make any assumptions on the robot's motion model, but still achieves significant speedups in planning time over heuristic-based A*. Our algorithm builds on the previously-developed algorithm for path planning with adaptive dimensionality to explicitly decrease the dimensionality of the search space in an adaptive manner. The algorithm plans in full dimensional state-space in regions of the environment, where there is a potential dynamic obstacle collision and in low-dimensional state-space elsewhere, thereby obtaining quicker planning times. We have proven that our approach returns feasible, collision-free paths in dynamic environments with bounds on solution cost sub-optimality. As shown in our results, we outperform full-dimensional planning algorithms such as HCA* by a substantial margin in tasks like navigation of non-holonomic robot in dynamic environments.

Note that the major assumption in this algorithm was that the trajectories of the dynamic obstacles in the environment are known. This is a strong assumption to make, especially in chaotic dynamic environments such as dense human crowds. Hence, we need to have access to an accurate long-horizon trajectory prediction algorithm that can be used in conjunction with the planning algorithm presented in this chapter. As new sensory information is obtained, the predictions are updated and the path is re-planned. In the next chapter, we will present a novel trajectory prediction algorithm in dense human crowds, which has high long-term prediction accuracy and works for various crowd settings.

Chapter 4

Modeling Cooperative Navigation in Dense Human Crowds

In this chapter, we will present a novel trajectory prediction algorithm for humans in dense crowds assuming we have access to their past trajectories. The presented algorithm captures human-human interactions such as joint collision avoidance and cooperation. More specifically, we develop an approach that models the joint distribution over future trajectories of all interacting agents in the crowd, through a local interaction model that we train using real human trajectory data. The interaction model infers the velocity of each agent based on the spatial orientation of other agents in his vicinity. During prediction, our approach infers the goal of the agent from its past trajectory and uses the learned model to predict its future trajectory. We demonstrate the performance of our method against a state-of-the-art approach on a public dataset. Results show that our model outperforms when predicting future trajectories for longer horizons. This chapter is adapted from our paper Vemula et al. [49] presented at ICRA 2017.

4.1 Introduction

There is an increasing need for robots to operate in the midst of human crowds. This requires the robot to be able to navigate through a crowd in a socially compliant way, i.e., the robot needs to collaboratively avoid collisions with humans and adapt its trajectories in a human predictable manner.

To date, the majority of existing works in the area of social navigation has focused on the prediction of individual motion patterns in the crowd to improve the navigation performance [50, 51, 52]. However, even in the case of perfect prediction, these approaches can lead to severely suboptimal paths [1]. The primary reason for such underperformance is that these approaches do not capture the complex and often subtle interactions that take place among humans in a crowd; that is, these approaches model each agent independently of the others. This observation leads to the insight that agents in crowds engage in joint collision avoidance.

Humans navigate through dense crowds by adapting their trajectories to those of other people in the vicinity. Figure 4.1 shows three examples of such behavior where people pass through, slow down, or go around when they are near other pedestrians. In order to learn

to navigate in a socially compliant way, it is key to capture such human-human interactions observed in a crowd.

Pioneering works by Helbing and Molnar [29], Hall [53] propose hand-crafted functions to model such interactions based on proximity. Such functions are, however, limited in the complexity of interactions that they can model and fail to generalize for crowded settings. Trautman et. al. Trautman and Krause [1] proposed an approach that explicitly models human-human and human-robot interactions to enable a robot to safely navigate in a dense crowd. The trajectories of the robot and the humans are jointly predicted with a hand-crafted potential term to model interactions.

Because the potential term is hand-crafted, it is possible that the robot trajectories generated may not resemble socially compliant human behavior. In this paper, we learn the interaction model from real-world pedestrian trajectories in order to predict human-like trajectories.

In safety-critical applications like robotics, where robots are expected to navigate safely among humans, we need to account for uncertainty in our predictions. Those approaches that learn interaction models but that do not deal with uncertainty as in Alahi et al. [23] can lead to over-confident predictions which could result in awkward and disruptive behavior. Our approach considers the uncertainty regarding intentions (or goals) of the pedestrians and results in accurate predictions.

The summary of our contributions in this chapter are as follows: We develop a new algorithm for enabling robots to move naturally through dense crowds. Following the key insight that agents in crowd engage in *joint collision avoidance*, we develop an approach that models the distribution over future trajectories of all interacting agents through a joint density model that captures the idea of *cooperative* planning, i.e., agents cooperating with each other to achieve their goals and avoid collisions. To capture collision avoidance behavior, we learn a local interaction model that encodes how agents move based on how populated their vicinity is from real human trajectory data. During prediction, our model infers the goal of an agent from its past trajectory and uses the learned model to predict its future trajectory. Finally, we demonstrate that our model is capable of predicting robot trajectories that are natural and human-like by reporting the experimental results on the ETH pedestrian video dataset [9].

In the remainder of this chapter, we will give an overview of relevant existing work in section 4.2. The notation and the problem is defined in section 4.3. Section 4.4 will describe our approach in detail and its evaluation on a real world dataset is presented in section 4.5. The results of the evaluation are discussed in section 4.6 and the conclusions are presented in section 4.7, along with directions for future work.

4.2 Related Work

4.2.1 Navigation in Uncertain Dynamic Environments

Common approaches to robot navigation in uncertain, populated environments typically compute a path to the goal without taking into account the collaborative collision avoidance behavior exhibited by humans. Most of the approaches instead rely on local reactive collision avoidance methods [4, 41]. Although these methods effectively avoid collisions with humans, the planned trajectories are usually suboptimal, compared to the trajectory a human would take for the same start and goal, and not socially compliant due to evasive maneuvers.

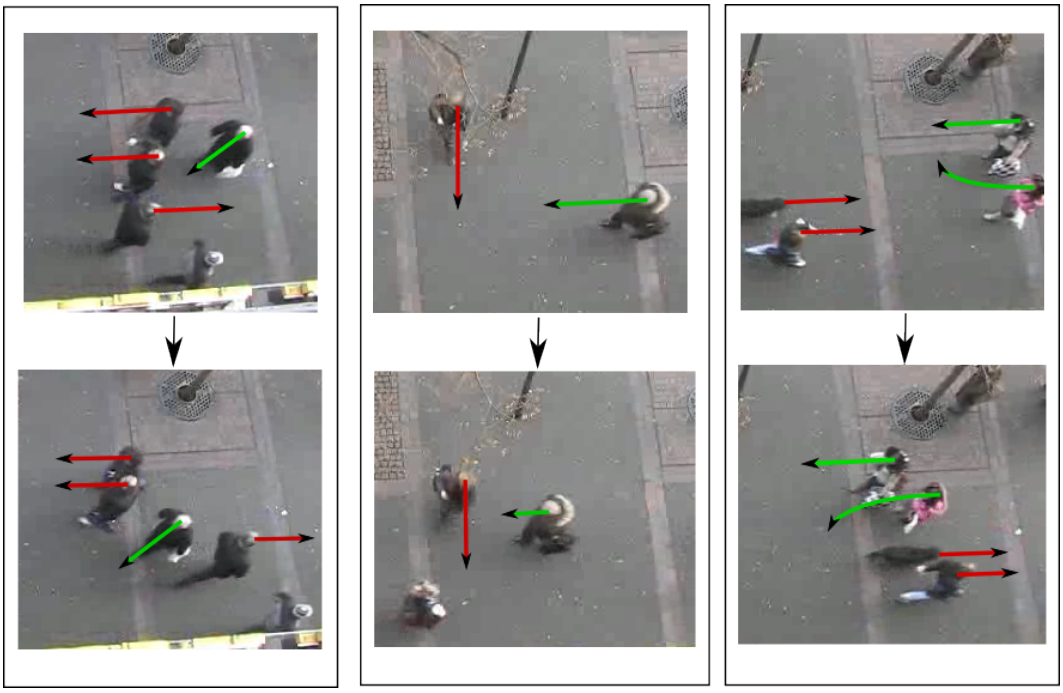


Figure 4.1: Examples of pedestrians exhibiting cooperative behavior. In each image, the velocity of the pedestrian is shown as an arrow where the length of each arrow represents the speed. (Left) The pedestrian (with green arrow) anticipates the open space between the other three (with red arrow) and doesn't slow down. (Middle) The pedestrian (with green arrow) slows down to allow the other pedestrian (with red arrow) to pass through. (Right) The two pedestrians (with green arrows) make way for the oncoming agents (with red arrow) by going around them.

Naively modeling unpredictability in the trajectories of humans, using linear models like Kalman filters, leads to increasing uncertainty that makes safe and efficient navigation difficult [1]. Several works have focused on controlling the uncertainty in these predictive models by developing accurate human motion models [50, 51], but these approaches do not account for interactions between humans and thus cannot model joint collision avoidance.

4.2.2 Modeling human interactions

The social forces model proposed by Helbing and Molnar [29] models motion of pedestrians in terms of forces that drive humans to reach a goal and to avoid obstacles. Subsequently, approaches have been proposed that use learning methods to fit the social forces model to observed crowd behavior [54, 55]. The social forces model has been shown to perform well in predicting trajectories in simulated crowds, but fails when it comes to predicting movements of pedestrians in real dense crowds as it uses a hand-crafted potential term based on distances, and doesn't learn human-human interactions from real data. Using a hand-crafted potential term results in a model that can capture simple interactions, like repulsion and attractions, but may fail to account for more complex crowd behavior. Treuille et al.

[56] models pedestrian motion behavior using dynamic potential fields that guide people to avoid collisions and move towards the goal, but this model does not explicitly account for interactions like cooperation between agents in the crowd.

Hall [53] introduces a theory on human proximity relationships which have been used in potential field methods to model human-human interactions [57, 58]. These models capture interactions to avoid collisions, but do not model human-robot cooperation. However, models of cooperation are necessary for safe and efficient navigation in dense crowds [1], because in cases where the crowd density is high, the robot believes there is no feasible path in the environment unless it accounts for cooperation from the crowd. Reciprocal Velocity Obstacles (RVO), [26], and Velocity Obstacles (VO), [59], account for interactions between agents by computing joint collision-free velocities assuming constant velocities and shared collision avoidance behaviors. However, these approaches cannot handle stochastic behavior in pedestrian motions and do not train the model from real observed data.

Trautman and Krause [1] proposed *Interacting Gaussian processes* (IGP) to explicitly model the human-robot cooperation. Similar to the work presented in this paper, IGP models the trajectories of all interacting agents jointly which results in a probabilistic model that can capture joint collision avoidance behavior. However, the IGP model assumes that the final destinations of all pedestrians are known, which is not the case in a realistic prediction task. Another drawback of IGP is the use of hand-crafted interaction potential term to model cooperative behavior which may result in robot trajectories that are not socially compliant. In this paper, we learn the interaction model from observations of real pedestrian trajectory data in the hope that we achieve more human-like and socially compliant trajectories.

The works of Kretzschmar et al. [18], Kuderer et al. [60] are also closely related to our work. These approaches explicitly model human-robot cooperation and jointly predict the trajectories of all agents, using feature-based representations. Unlike our proposed approach, they use *maximum entropy inverse reinforcement learning* (IRL) to learn an interaction model from human trajectory database using carefully designed features such as clearance, velocity, or group membership. However, their approach has been tested in scripted environments with no more than four humans. In our work, we deal with crowded scenarios with an average of six humans in a single scene. Very recently, Pfeiffer et al. [61] have extended the maximum entropy approach to unseen and unstructured environments by using a receding horizon motion planning approach.

4.2.3 Trajectory prediction

A large body of works exist in the domain of computer vision and video surveillance that deal with predicting motion of people in videos, that are relevant to our work. Joseph et al. [19], Kim et al. [62] learn motion patterns using Gaussian processes and cluster human trajectories into these motion patterns. But these approaches ignore human-human interactions. IRL has also been used in the domain of activity forecasting to predict human paths in static scenes, [20] and more recently, Alahi et al. [23] used Long Short-Term Memory networks (LSTM) to jointly reason across multiple agents to predict their trajectories in a scene. However, most of these approaches have been used in the context of prediction and have not been extended to navigation.

4.3 Problem Definition

In this section, we will lay out the notation followed in the rest of the chapter, briefly explain how planning reduces to inference using joint density and formally define the problem of modeling the joint density.

4.3.1 Notation

We follow the notation of Trautman and Krause [1]. Let the index $i \in \{1, 2, \cdots, N\}$ specify the agent, where N is the number of individuals in the crowd and $i = R$ indicates the robot. The trajectory of agent $i \in \{R, 1, 2, \cdots, N\}$ is given by $\mathbf{f}^{(i)} = (f_1^{(i)}, f_2^{(i)}, \cdots, f_T^{(i)})$, where T is the length of the trajectory and $f_t^{(i)} = (x_t^{(i)}, y_t^{(i)}) \in \mathbb{R}^2$ is the location of agent i at time-step t. The observed locations of pedestrian i until time-step t is denoted as $\mathbf{z}_{1:t}^{(i)} = (z_1^{(i)}, z_2^{(i)}, \cdots, z_t^{(i)})$. We denote the set of all pedestrian trajectories by $\mathbf{f} = \{\mathbf{f}^{(i)}\}_{i=1:N}$, the robot trajectory by $\mathbf{f}^{(R)}$, and the set of all pedestrian observations until time-step t by $\mathbf{z}_{1:t} = \{\mathbf{z}_{1:t}^{(i)}\}_{i=1:N}$. We assume a fixed number of goals g in the environment are known and denote the set of goals by $\mathbf{G}$.

4.3.2 Planning using the joint density

Following the assumption that people engage in a joint collision avoidance when moving through a dense crowd as in Trautman and Krause [1], Helbing and Molnar [29], the robot does not only have to respond to the observed trajectories of the pedestrians, but also has to account for the adaptive behavior of the humans.

To capture this cooperative behavior, it has been suggested by Trautman and Krause [1] to use the joint density of both the robot and the crowd, denoted by $P(\mathbf{f}^{(R)}, \mathbf{f} | \mathbf{z}_{1:t})$. Planning the path for the robot using this density corresponds to finding the maximum-a-priori (MAP) assignment for the following posterior:

$$(\mathbf{f}^{(R)}, \mathbf{f})_* = \underset{\mathbf{f}^{(R)}, \mathbf{f}}{\arg\max}\, P(\mathbf{f}^{(R)}, \mathbf{f} | \mathbf{z}_{1:t}). \tag{4.1}$$

4.3.3 Problem

In this work, we assume that at each time-step t, we receive the observation $\mathbf{z}_t$ of the locations of all agents in the crowd. Given the current and past observations $\mathbf{z}_{1:t}$, we tackle the problem of estimating the joint posterior distribution of the future trajectories of all agents. Formally, we seek to model the density given by,

$$P(\mathbf{f}^{(R)}, \mathbf{f} | \mathbf{z}_{1:t}).$$

Planning the robot's trajectory then corresponds to taking the MAP assignment for $\mathbf{f}^{(R)}$ and executing it until the next observation is received. At time-step $t + 1$, we receive a new observation $\mathbf{z}_{t+1}$ and update the above joint posterior density to $P(\mathbf{f}^{(R)}, \mathbf{f} | \mathbf{z}_{1:t+1})$. This process is repeated until the robot reaches its destination. In contrast to Trautman and Krause [1], who tackle a similar problem, we aim to predict more natural and human-like robot trajectories by learning the model from pedestrian trajectory data.

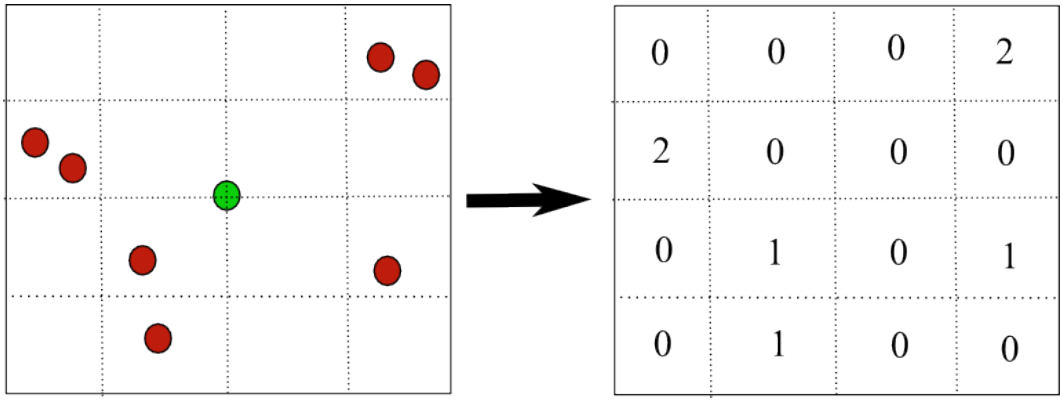


Figure 4.2: Occupancy grid construction. (Left) A configuration of other agents (red) around the current agent (green). (right) 4x4 occupancy grid is constructed using the number of agents in each grid cell

4.4 Approach

We exploit the observation that humans navigating in dense crowds adapt their trajectories based on the presence of agents in their vicinity [23]. We first explain the construction of *occupancy grids*, which account for the presence of other agents within an agent's local neighborhood (Section 4.4.1). We formulate the problem of learning the social interaction model as a Gaussian process regression problem, where we predict the agent's velocities at each time-step as a function of their occupancy grids and intended goal. Given the preprocessed training trajectories (Section 4.4.2), we train the GP model by maximizing its marginal likelihood to learn the hyperparameters of the kernel (Section 4.4.3). At prediction time, we use the learned model to infer the goal of each agent and jointly predict future trajectories of all interacting agents in the crowd (Section 4.4.4).

4.4.1 Constructing occupancy grids

To capture the local interactions of an agent with its neighbors, we construct an occupancy grid for each agent i at each time-step t that is similar to the social pooling grid used in Alahi et al. [23]. The occupancy grid is constructed by discretizing the neighborhood of the agent's current location into a spatial grid of size $M \times M$. We then count the number of surrounding agents within each grid cell. Formally, we define occupancy grid as a vector of length M^2 given by,

$$\mathbf{O}_t^{(i)}(a + M(b-1)) = \sum_{j \in \mathcal{N}^{(i)}} \mathbb{I}_{a,b}[x_t^{(j)} - x_t^{(i)}, y_t^{(j)} - y_t^{(i)}]. \tag{4.2}$$

$\mathcal{N}^{(i)}$ denotes the set of all agents $j \neq i$, who are within the neighborhood of agent i. The indicator function $\mathbb{I}_{a,b}[x, y]$ defines if (x, y) is located in the (a, b) cell of the occupancy grid. In the remainder of the paper, $\mathbf{O}^{(i)}$ and $\mathbf{O}$ will denote the set of all occupancy grids at every time-step of an agent i and that of all agents, respectively.

4.4.2 Preparing training data for learning

Before training our model, we preprocess the training data $\mathbf{f}$, which correspond to the trajectories of all agents. First, we construct occupancy grids $\mathbf{O}^{(i)}$ and $\mathbf{O}$ along $\mathbf{f}$, as described in Section 4.4.1. Second, we process all the trajectories to obtain the velocities of agents at each time-step $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$. Third, since we have access to the entire trajectory at training, we can compute the goals of all the pedestrians. Note that this information about the true goals is used only during training and is not assumed in the prediction phase. After this preprocessing, we have $(\mathbf{O}, \frac{\Delta \mathbf{x}}{\Delta t}, \frac{\Delta \mathbf{y}}{\Delta t})$ for all agents at every time-step and their corresponding goals $\{g^{(i)}\}$.

4.4.3 Training the local interaction model

To learn across different pedestrians traversing in different regions of the environment, we model the velocity of an agent at a specific time-step as a function of their intended goal and their occupancy grid at that time-step. Formally, we seek to estimate the distribution $P(\frac{\Delta \mathbf{x}}{\Delta t}|\mathbf{O}, g)$ and $P(\frac{\Delta \mathbf{y}}{\Delta t}|\mathbf{O}, g)$, for each goal g in $\mathbf{G}$, from the training data obtained in Section 4.4.2.

We start by modeling the interactions as a Gaussian Process (GP) regression problem, where the noisy data to be interpolated is $(\mathbf{O}, \frac{\Delta \mathbf{x}}{\Delta t})$ and $(\mathbf{O}, \frac{\Delta \mathbf{y}}{\Delta t})$. Note that we use a separate GP for each goal g to learn the mapping from occupancy grids to velocities. We use a *squared exponential automatic relevance determination* (SE-ARD) kernel (with additive noise) for these GPs [63]. The SE-ARD kernel learns a different lengthscale for each input dimension, which in our problem are the dimensions of the occupancy grid vector, i.e., the grid cells. Since, the kernel can learn the relevance of each input dimension by learning the lengthscale parameter [63] (dimensions with large lengthscales are less relevant than those with small lengthscales), the kernel will capture the relevance of each grid cell for a specific goal and effectively ignore irrelevant cells.

The SE-ARD kernel with additive noise is given by,

$$K_S(\mathbf{O}, \mathbf{O}') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{M^2}\frac{(\mathbf{O}(d) - \mathbf{O}'(d))^2}{\ell_d^2}\right) + \sigma_n^2 \delta(\mathbf{O}, \mathbf{O}'), \tag{4.3}$$

where $\delta(\mathbf{O}, \mathbf{O}') = 1$ if $\mathbf{O}$ is equal to $\mathbf{O}'$ and zero otherwise, and $\mathbf{O}(d)$ is the value of the d^{th} dimension in the vector $\mathbf{O}$. The hyperparameters of this kernel are σ_f (signal variance), $\{\ell_d\}_{d=1}^{M^2}$ (lengthscales) and σ_n (noise variance).

This construction results in a total of $2G$ GPs because there is a pair of GPs (in x- and y-direction) for each of the G goals; thus, we have $2G$ sets of hyperparameters to be learned. We denote the set of hyperparameters corresponding to the GP associated with goal g by Θ_x^g and Θ_y^g. To learn the hyperparameters Θ_x^g, we isolate the tuples $B_g = \{(\mathbf{O}^{(i)}, \frac{\Delta \mathbf{x}}{\Delta t}^{(i)})\}_i$ from training data, corresponding to the set of pedestrians i whose goal is g, and maximize the log marginal likelihood of the GP [63] given by,

$$\log P\left(\frac{\Delta \mathbf{x}}{\Delta t}|\mathbf{O}\right) = -\frac{1}{2}\frac{\Delta \mathbf{x}}{\Delta t}^T K_S(\mathbf{O},\mathbf{O})^{-1}\frac{\Delta \mathbf{x}}{\Delta t} - \frac{1}{2}\log|K_S(\mathbf{O},\mathbf{O})| - \frac{n_g}{2}\log 2\pi, \tag{4.4}$$

where $\frac{\Delta \mathbf{x}}{\Delta t}$ and $\mathbf{O}$ are vectors constructed by concatenating elements of B_g, and n_g is the number of elements in $\frac{\Delta \mathbf{x}}{\Delta t}$.

We can learn Θ_y^g and all the other sets of hyperparameters for all goals $g \in \mathbf{G}$ in a similar fashion.

4.4.4 Prediction

During prediction we are given an unseen crowd with N_p pedestrians, a robot, and their observations $\mathbf{z}_{1:t}$ until time t. Our task is to predict their trajectories $\mathbf{f}$ and $\mathbf{f}^{(R)}$ for H time-steps into the future, using the learned model. We cannot directly use the GP predictive distribution because we do not know the goals of the pedestrians during prediction. Note that we know the goal of the robot $g^{(R)}$ since it is user-defined.

Infer goal of a pedestrian

Given observations $\mathbf{z}_{1:t}^{(i)}$ of pedestrian i until time t and the set of goals $\mathbf{G}$, we seek to infer the goal $g^{(i)} \in \mathbf{G}$ of the pedestrian. We assume a uniform prior $P(g^{(i)})$ over all goals, in the absence of any observations for agent i (A more informative prior over the goals can be found by analyzing the environment). Hence, we have:

$$P(g^{(i)}|\mathbf{z}_{1:t}^{(i)}) = \frac{P(\mathbf{z}_{1:t}^{(i)}|g^{(i)})P(g^{(i)})}{P(\mathbf{z}_{1:t}^{(i)})} \propto P(\mathbf{z}_{1:t}^{(i)}|g^{(i)}). \tag{4.5}$$

That is, we evaluate the likelihood that the observation sequence $\mathbf{z}_{1:t}^{(i)}$ is true conditioned on the fact that $g^{(i)}$ is the goal of agent i. Similar approaches have been explored in Kitani et al. [20] and Ziebart et al. [64], for inferring destination of an agent given its previous path.

To compute the likelihood, we first compute, from the observations $\mathbf{z}_{1:t}$, the velocities $\{(\frac{\Delta \mathbf{x}}{\Delta t}, \frac{\Delta \mathbf{y}}{\Delta t})\}_{1:t-1}$ and the occupancy grids $\mathbf{O}_{1:t-1}$ of all pedestrians at each time-step until $t-1$. For each possible goal $g \in \mathbf{G}$, we take the corresponding set of trained hyperparameters Θ_x^g and Θ_y^g, and evaluate the log marginal likelihood of the GP for each agent i (using equation 4.4). Hence, for each agent i and each goal $g^{(i)} \in \mathbf{G}$, we obtain the likelihood that its observed data $\mathbf{z}_{1:t}^{(i)}$ is generated from the GP conditioned on the goal $g^{(i)}$. Normalizing the likelihoods across all goals, we get the likelihood $P(\mathbf{z}_{1:t}^{(i)}|g^{(i)})$ for every goal $g^{(i)} \in \mathbf{G}$.

Predicting future trajectories

Now that we have a distribution over the goals $g^{(i)}$ for all agents i in the crowd, we can use the trained model to predict future locations. The joint posterior density can be decomposed as

$$P(\mathbf{f}^{(R)}, \mathbf{f}|\mathbf{z}_{1:t}) = \sum_{\mathbf{g}} P(\mathbf{f}^{(R)}, \mathbf{f}|\mathbf{g}, \mathbf{z}_{1:t})P(\mathbf{g}|\mathbf{z}_{1:t}), \tag{4.6}$$

where $\mathbf{g} = \{g^{(i)}\}_{i=R,1:N}$ are the goals of all agents including the robot. We can assume that the goals of the pedestrians are independent of each other (and that we know the goal of the robot with certainty) given their respective observations. Then, we can write the distribution of a goal given a history of observations as:

$$P(\mathbf{g}|\mathbf{z}_{1:t}) = \prod_{i=1}^{N} P(g^{(i)}|\mathbf{z}_{1:t}^{(i)}), \tag{4.7}$$

where $P(g^{(i)}|\mathbf{z}_{1:t}^{(i)})$ is given by equation 4.5. We approximate the joint distribution $P(\mathbf{f}^{(R)}, \mathbf{f}|\mathbf{g}, \mathbf{z}_{1:t})$ by using the velocities and occupancy grids obtained from observations $\mathbf{z}_{1:t}$ (as done in Section 4.4.4),

$$P(\mathbf{f}^{(R)}, \mathbf{f}|\mathbf{g}, \mathbf{z}_{1:t}) \approx P(\mathbf{f}^{(R)}, \mathbf{f}|\{\frac{\Delta\mathbf{x}}{\Delta t}, \frac{\Delta\mathbf{y}}{\Delta t}\}_{1:t-1}, \mathbf{O}_{1:t}, \mathbf{g}). \tag{4.8}$$

The predictions for different agents are coupled through the occupancy grid which contains the configuration of other agents around each agent locally. This enables our model to capture local interactions, like joint collision avoidance and cooperation.

Since the task is to predict the future locations of all agents for the next H time-steps, $\mathbf{f}^{(i)}$ suffices to represent the next H locations of agent i after time t, in addition to previous locations.

Multi-step prediction

Future locations can be predicted using the learned model from Section 4.4.3. For each agent i, we fit a separate pair of GPs (with the learned hyperparameters Θ_x^g, Θ_y^g for goal g) to the observed tuples $(\mathbf{O}_{1:t}^{(i)}, \{\frac{\Delta\mathbf{x}}{\Delta t}\}_{1:t-1}^{(i)})$ and $(\mathbf{O}_{1:t}^{(i)}, \{\frac{\Delta\mathbf{y}}{\Delta t}\}_{1:t-1}^{(i)})$. Using their corresponding GPs, each agent can predict their velocities and compute the location for the next time-step by adding it to the current location.

This can be done exactly for time $t+1$, i.e., we can predict $(\frac{\Delta\mathbf{x}}{\Delta t})_{t+1}^{(i)}$ and $(\frac{\Delta\mathbf{y}}{\Delta t})_{t+1}^{(i)}$ for each agent i, since we know the value of the occupancy grid at time t, $\mathbf{O}_t^{(i)}$. But for future time-steps, we need to estimate the occupancy grid at the previous time step using previous predictions. Instead of computing the distribution over future locations in an exact form (which can be extremely difficult), we use Monte Carlo sampling to approximate the distribution as shown in Algorithm 2.

At time $t+1$, we compute the GP predictive distribution [63] for each $(\frac{\Delta\mathbf{x}}{\Delta t})_{t+1}^{(i)}$ and $(\frac{\Delta\mathbf{y}}{\Delta t})_{t+1}^{(i)}$ (line 2). We then proceed to sample S points from each of these distributions (line 6), and estimate S samples for the location $\mathbf{f}_{t+1}^{(i)}$ for all agents i (line 7). These sets of samples approximate the distribution $P(\mathbf{f}_{t+1}^{(R)}, \mathbf{f}_{t+1}|\{\frac{\Delta\mathbf{x}}{\Delta t}, \frac{\Delta\mathbf{y}}{\Delta t}\}_{1:t-1}, \mathbf{O}_{1:t}, \mathbf{g})$.

Since, for each sample, we have locations of all agents at time $t+1$, we can compute occupancy grids $\mathbf{O}_{t+1}^{(i)}$ for each agent (line 9). Thus, we get S samples for $\mathbf{O}_{t+1}$. Now, to estimate location at time $t+2$, we compute the mean of the S samples to get the set of occupancy grids, $\mathbf{O}_{t+1}$ (line 10). Using the mean, we predict the velocities at time $t+2$ (line 12), and repeat the above process until H time-steps into the future. At every time-step t' ($t' \geq t+1, t' \leq t+H$), we get a set of S samples corresponding to the locations $\mathbf{f}_{t'}$ that approximate the distribution

$$\{\mathbf{f}_{t'}^{(R)}, \mathbf{f}_{t'}\}_{j=1:S} \approx P(\mathbf{f}_{t'}^{(R)}, \mathbf{f}_{t'}|\{\frac{\Delta\mathbf{x}}{\Delta t}, \frac{\Delta\mathbf{y}}{\Delta t}\}_{1:t-1}, \mathbf{O}_{1:t}, \mathbf{g}).$$

As we let the value of S grow, we get a better approximation.

Algorithm 2 Multi-step prediction through Sampling

for each agent i **do**
  Compute distributions of $(\frac{\Delta \mathbf{x}}{\Delta t})^{(i)}_{t+1}$, $(\frac{\Delta \mathbf{y}}{\Delta t})^{(i)}_{t+1}$ given $\mathbf{O}^{(i)}_t$
end for
for $t' = t+2 \rightarrow t+H$ **do**
  for each agent i **do**
    Sample S points from distributions of $(\frac{\Delta \mathbf{x}}{\Delta t})^{(i)}_{t'-1}$, $(\frac{\Delta \mathbf{y}}{\Delta t})^{(i)}_{t'-1}$
    Compute S estimates for $\mathbf{f}^{(i)}_{t'}$ from sampled velocities
  end for
  Compute S samples for $\mathbf{O}_{t'}$ from estimates of $\mathbf{f}_{t'}$
  Set $\mathbf{O}_{t'}$ to be the mean of the S samples from above
  for each agent i **do**
    Compute distributions of $(\frac{\Delta \mathbf{x}}{\Delta t})^{(i)}_{t'}$, $(\frac{\Delta \mathbf{y}}{\Delta t})^{(i)}_{t'}$ given $\mathbf{O}^{(i)}_{t'}$
  end for
end for
return $\{\{\mathbf{f}^{(R)}_{t'}, \mathbf{f}_{t'}\}_{j=1:S}\}_{t'=t+1:t+H}$

4.5 Evaluation

In this section, we will describe the setup for the experiments and present results in comparison to the baseline algorithm.

4.5.1 Setup

We evaluate our model on a publicly available human-trajectory dataset released by ETH, [9]. The dataset contains a video recorded from above a busy doorway of a university building with the pedestrian trajectories tracked and annotated. This video contains scenes with real world crowded settings, hundreds of trajectories and high crowd density. An example snapshot from the video (with goals marked) is shown in figure 4.3. Each time-step in the video is six frames long and amounts to 0.4 seconds. The average trajectory length of a pedestrian is 25 time-steps. The total number of pedestrians in the video is 360 and there are four goals in the environment. The resolution of the video is 640×480. Each pixel in the video frame corresponds to 0.042 metres (slightly varies across the frame, as the camera is angled and not exactly top-down).

We evaluate our model by choosing a pedestrian in the crowd randomly as our robot and use his start and goal state to plan a path through the crowd. The resulting path is compared to the true path that the pedestrian has taken in the video. Comparing the true path and the predicted path gives us a evaluation of how closely our prediction resembles human-like behavior. A similar evaluation was done in Trautman and Krause [1].

We compare our approach against IGP [1], as it deals with the same problem of jointly modeling trajectories of robot and pedestrians, and has shown good results in real robot navigation scenario among dense crowds [10, 11]. We will use both our approach and IGP to predict the path until H time-steps into the future and compare it with the true trajectory.

Figure 4.3: Example snapshot of the dataset with goals indicated by red dots

Note that the original IGP needs to know the true final destination of each pedestrian at prediction time, which would give it an unfair advantage over our algorithm. Hence, we use a variant of IGP which doesn't need the true final destinations and use that in our comparison. At prediction time, we compute the average heading of the pedestrian for the last 5 time-steps and estimate the goal location in the computed heading direction. The estimated goal is used in the original IGP algorithm in place of the true final destination of the pedestrian.

To compare the path predicted by the two algorithms and the true path of the pedestrian, we consider two metrics:

1. *Average displacement error*: Introduced in Pellegrini et al. [9], this metric computes the mean squared error over all estimated points at each time-step in the predicted trajectory and the true trajectory.

2. *Final displacement error*: Introduced in Alahi et al. [23], this metric computes the mean distance between the final predicted location after H time-steps and the true location after H time-steps, where H is our prediction horizon.

4.5.2 Model Parameters

We construct occupancy grids around each agent of size 4×4 (i.e., $M = 4$) covering a space of 80×80 pixels in the video. In each video, we train the model on the trajectories of the first 50 pedestrians. At prediction time, we chose a scenario with 11 pedestrians (from the remaining part of the video, not used in training), one of whom is used as a robot in our model. We predict the future locations for a range of prediction horizons $H = 1, 2, 5, 10, 20$ time-steps. If the path of the pedestrian (or robot) ends in less than H time-steps, we will predict only until the end of his path. For multi-step prediction, we use $S = 100$ samples

Figure 4.4: Example prediction by our model. For each pedestrian, we predict his future locations (which are plotted) for the next 5 time-steps. The bottom set of pedestrians are progressing towards a goal at the top centre of the image, but they go around the other set of pedestrians making way for them cooperatively.

to approximate the distribution over future locations. We implement the Gaussian process regression model using the GPML toolbox, [63].

4.5.3 Results

To test the prediction accuracy of both approaches, we have chosen scenarios with 11 pedestrians in the crowd where crowd density is high and some pedestrians head into and through the crowd. To get an unbiased estimate, we chose 5 such scenarios in the video. In each scenario, we assume each pedestrian to be the robot, one at a time, and compute their average displacement error and final displacement error, averaged over all time-steps. This results in 11 sets of error values and we compute the average over all sets to give the mean errors over all pedestrians. We repeat the experiment for different H values to get both short-range and long-range prediction accuracies of both approaches. The results are averaged over all 5 scenarios and are presented in Table 4.1. Note that the errors are listed in pixels.

In Figure 4.4, we show an example scene where our approach predicts cooperative behavior. The set of pedestrians going up give way to the set of pedestrians going down. To visualize what our local interaction model (from section 4.4.3) learned, we give it some ex-

Table 4.1: Prediction errors (in pixels) on the dataset for IGP and our approach

Metric	Prediction horizon (H)	IGP	Our Approach
Avg. Disp. Error	1	3.42	4.42
	2	5.66	6.14
	5	15.75	12.09
	10	21.59	21.52
	20	41.51	34.63
Final Disp. Error	1	3.42	4.42
	2	7.12	7.78
	5	23.18	19.77
	10	38.75	36.25
	20	67.41	54.2

ample occupancy grids and goals, and observe the predicted velocities. Figure 4.5 shows that the model learns collision avoidance as it predicts velocities away from grid cells which are occupied and towards unoccupied grid cells. When an agent's vicinity is heavily populated in the direction of its goal, the magnitude of predicted velocity is very low, i.e., the agent moves slowly. If instead, its vicinity is populated in the opposite direction of its goal, the velocity of the agent doesn't get affected by the surrounding agents (as they are not obstructing its path).

To verify this observation, we have examined the values of the learned hyperparameters of the SE-ARD kernel. The lengthscales for grid cells that are not in the direction of the pedestrian's intended goal, are given high values, thus reducing their relevance in the velocity prediction. For example, in Figure 4.5 the lengthscales for the bottom grid cells in the left bottom occupancy grid, are set to values higher than 7 whereas the lengthscales for the top grid cells in the same grid are set to values lower than 1. This shows that our model learns how neighbors affect a pedestrian's path based on their relative spatial orientation.

4.6 Discussion

From the results presented in Table 4.1, we can observe that our approach performs better than IGP at predicting human trajectories for longer prediction horizons and worse for shorter horizons. This is mainly because IGP models trajectories directly by predicting future locations based on previously observed locations. This results in very accurate predictions in situations where there are no surrounding agents (and hence, no interactions) and for shorter prediction horizons, as it extrapolates the trajectory to future time-steps smoothly. Our model, on the other hand, models velocities at each time-step and needs to estimate the future location based on velocity predictions for the previous time-step. Thus, for shorter prediction horizons, our model has a higher variance associated with predicted locations. But for longer horizons, our model has higher accuracy in prediction as it reasons about the intended goal of the agent and captures local interactions at each time-step. IGP fails at longer horizons as the smooth extrapolation, coupled with the handcrafted interaction potential term, is unable to account for all the interactions and cooperative behavior among dense crowds. More importantly, our approach has a higher performance than IGP because it learns the local interaction model from real human trajectory data whereas the interaction model in IGP is hand-crafted.

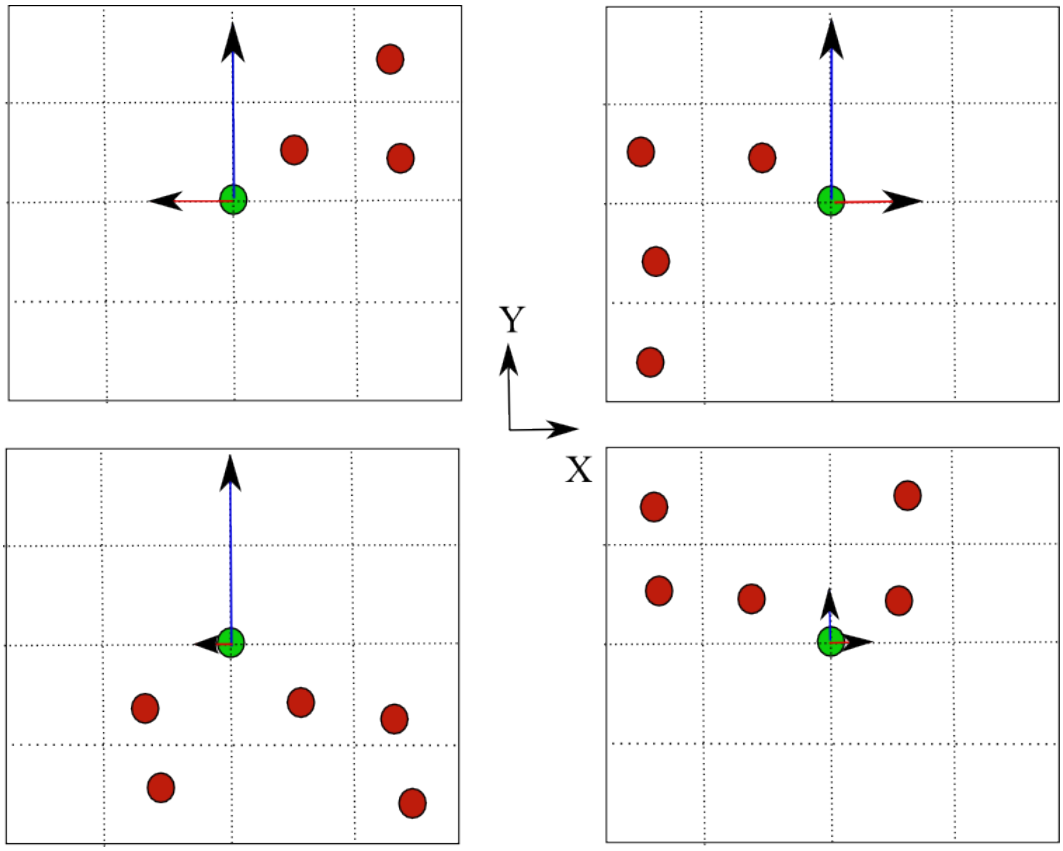


Figure 4.5: Velocities predicted by our trained model for example occupancy grids. In each case, the goal of the pedestrian is right above in the Y-direction. Predicted mean y-velocity is shown in blue and predicted mean x-velocity is shown in red.

Accurate predictions for longer horizons is important in social navigation as it results in a more globally optimal behavior. In cases where the prediction is accurate in a short horizon but poor for longer horizons, the resulting paths are locally optimal and can potentially lead to a non-socially compliant and reactive behavior.

Upon careful examination of predictions of our approach in crowded scenarios, we observed that it learns the behavior of slowing down (see bottom right of Figure 4.5) when its vicinity is heavily populated, which is a common behavior among humans. Also, as observed from the values of the learned lengthscales for the SE-ARD kernel, our model learns how humans decide their velocity based on the relative spatial configuration of other agents in their neighborhood. As shown in Figure 4.5, our trained local interaction model captures collision avoidance based on an agent's occupancy grid by learning from human trajectory data without any hand-crafted potential term.

Although we present results for predicting trajectories of every agent in the crowd, this approach can be extended to robot navigation by treating the robot as an agent in the crowd. Planning the path of the robot in this model reduces to inference in the joint density as shown in Section 4.3.2. The resulting path taken by the robot is the most likely path pre-

dicted according to the learned model. Recent work by Pfeiffer et al. [61] has shown that as long as pedestrians interact with the robot naturally (as one of them), such an interaction-aware modeling approach is significantly better than a reactive approach.

4.7 Summary

In this chapter, we presented a new approach to modeling cooperative behavior among humans in dense crowds. While most existing approaches use hand-crafted models to capture interactions among humans in the crowd, we take a data-driven approach and learn an interaction model from real human trajectory data. We propose a nonparametric statistical model that uses Gaussian processes to model velocities of agents in the crowd as a function of how populated their vicinity is. We show how our model can be used to predict future trajectories of pedestrians and compute the path of a robot through a dense crowd. The future trajectories are computed using a Monte Carlo sampling approach to multi-step prediction. Lastly, the efficacy of our approach is demonstrated by predicting trajectories of agents in a real world pedestrian dataset. Our results show that the model captures important aspects of human crowd behavior such as cooperative navigation and collision avoidance.

The drawback of our approach is the assumption of known goals in the environment. This restricts the generalizability of the approach to previously seen environments and a separate model needs to be trained for a new environment.

Chapter 5

Conclusion

In this chapter, we will summarize the contributions of the thesis and lay out future directions of research in the area of robot navigation in dynamic environments.

5.1 Summary

In this thesis, we have considered the problem of mobile robot navigation in dynamic environments. In particular, we explored two questions:

- Given an accurate model of world dynamics, how do you efficiently obtain safe, feasible and bounded sub-optimal paths for mobile robots in dynamic environments?
- How do you model complex dynamics such as cooperative behavior in dynamic environments, specifically human crowds?

We addressed the first question in Chapter 3, where we proposed a heuristic based graph search planning algorithm that uses adaptive dimensionality to only consider time dimension in the search space corresponding to regions where potential dynamic obstacle collisions can occur. This reduction in search space lends the approach great speed-ups in planning time over traditional approaches such as HCA*, without sacrificing safety, dynamic feasibility and bounded sub-optimality of the solution. The proposed planning algorithm has been shown to outperform HCA* in densely populated environments with a large number of dynamic and static obstacles, without making any assumption regarding the robot's motion model.

Chapter 4 addresses the second question of modeling environment dynamics by proposing a novel statistical model that aims to capture the complex and subtle interactions between humans in a dense crowd. Following previous approaches, we model the joint distribution of trajectories of all interacting agents without sacrificing scalability by introducing the abstraction of occupancy grids. We decompose the joint distribution by coupling predictions for individual trajectories through the occupancy grid thereby, capturing interactions that are dependent on relative distance and orientation between agents. The proposed model has shown to outperform IGP over long prediction horizons and learns complex behavior such as joint collision avoidance, slowing down and other cooperative behavior.

In addition to the aforementioned novel contributions of this thesis, Chapter 2 presents a brief survey of past works that have addressed similar challenges. The survey presents a good introduction for future researchers in this field to understand what has been done in the past and what challenges remain in order to make robot navigation in dynamic environments more safe and efficient.

5.2 Future Work

In this section, we present a few directions for future research to extend the proposed solutions in this thesis.

5.2.1 Path Planning in Dynamic Environments

The proposed planning algorithm in Chapter 3 has only been tested in simulations. As a future work, its performance can be verified on a mobile robot navigating in an environment with scripted dynamic obstacle trajectories. This can be taken further by using the planner in conjunction with a predictive model of the environment (such as the one in Chapter 4). Every time new sensor information is received, the model can be updated and a new path can be computed by re-planning using the proposed planner.

Another interesting direction would be to make the planner incremental so that search information from previous iterations can be re-used in subsequent iterations. Currently, the planner starts from scratch at the start of each iteration and does not reuse search tree from previous iterations. Extending it to be incremental lends itself naturally to this algorithm as it is iterative in nature and a large portion of the search tree remains the same across iterations. Hence, large speedups can be achieved in planning time making the algorithm more efficient and suitable for frequent re-planning.

Finally, most statistical models for world dynamics result in predictions with uncertainty associated with them. An extremely useful future direction would be to extend the proposed planning algorithm to account for the uncertainty, along with the prediction, to obtain probabilistic guarantees on safety of the planned path (similar work is done in Kushleyev and Likhachev [46]). We could also use this measure of uncertainty to obtain a good estimate of how frequently one should re-plan the path to ensure these probabilistic guarantees on safety.

5.2.2 Modeling Navigation in Dynamic Environments

The problem of modeling complex interactions in dynamic environments, such as human crowds, remains an unsolved problem. Our proposed statistical model in Chapter 4 takes a small step towards modeling cooperative behavior exhibited in human crowds resulting in more accurate predictions. As a future work, the model can be validated and verified on a real robot placed in a dense human crowd. The task would be, given a start and goal location, the robot should be able to navigate safely and efficiently through the crowd.

Currently, the model doesn't account for static obstacles which play a very important role in modeling navigation behavior. An interesting future direction would be to explore ways to account for both the dynamic agents and static obstacles in the environment, while predicting future trajectories. Another important drawback of our approach is the assumption of known goals in the environment. This restricts the generalizability of the approach

to previously seen environments and a separate model needs to be trained for a new environment. A future direction would be to move away from this assumption and achieve accurate long-term predictions in completely unknown environments.

Finally, our model only uses the positions and velocities as features to learn cooperative behavior from data. We could extend the model using temporal representation learning methods such as deep recurrent neural networks, to learn a latent representation for trajectories that encode high-level time varying information. This would enable the model to capture more complex and subtle interactions between agents in a dynamic environment.

Appendix A

Code and Publications

A.1 Code

- Code for the planning algorithm proposed in Chapter 3, with dataset : https://github.com/vvanirudh/sbpl_dynamic_adaptive_planner
- Code for the statistical model proposed in Chapter 4 : https://github.com/vvanirudh/occupancy-IGP

A.2 Publications

- The work presented in Chapter 3 was presented at SoCS 2016.

  Vemula, Anirudh, Katharina Muelling, and Jean Oh. "Path Planning in Dynamic Environments with Adaptive Dimensionality." In Ninth Annual Symposium on Combinatorial Search. 2016.

- The work presented in Chapter 4 was presented at ICRA 2017.

  Vemula, Anirudh, Katharina Muelling, and Jean Oh. "Modeling Cooperative Navigation in Dense Human Crowds." In International Conference on Robotics and Automation. 2017.

Appendix B

Public Datasets

- ETH Dataset used for evaluation in Chapter 4 can be found here : http://www.vision.ee.ethz.ch/en/datasets/

Bibliography

[1] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803, 2010.

[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[3] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation, 1999*, volume 1, pages 341–346. IEEE, 1999.

[4] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artif. Intell.*, 114:3–55, 1999.

[5] Sebastian Thrun, Michael Beetz, Maren Bennewitz, Wolfram Burgard, Armin B. Cremers, Frank Dellaert, Dieter Fox, Dirk Hähnel, Charles R. Rosenberg, Nicholas Roy, Jamieson Schulte, and Dirk Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *I. J. Robotics Res.*, 19:972–999, 2000.

[6] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[7] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[8] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[9] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, 2009.

[10] Peter Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2153–2160. IEEE, 2013.

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

[12] Frank Hoeller, Dirk Schulz, Mark Moors, and Frank E Schneider. Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1260–1265. IEEE, 2007.

[13] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[14] Sujeong Kim, Stephen J Guy, Wenxi Liu, David Wilkie, Rynson WH Lau, Ming C Lin, and Dinesh Manocha. Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, page 0278364914555543, 2014.

[15] Rachel Kirby, Reid Simmons, and Jodi Forlizzi. Companion: A constraint-optimizing method for person-acceptable navigation. In *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 607–612. IEEE, 2009.

[16] Beomjoon Kim and Joelle Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8(1):51–66, 2016.

[17] Masahiro Shiomi, Francesco Zanlungo, Kotaro Hayashi, and Takayuki Kanda. Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model. *International Journal of Social Robotics*, 6(3):443–455, 2014.

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

[19] Joshua Joseph, Finale Doshi-Velez, Albert S Huang, and Nicholas Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011.

[20] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.

[21] Matthias Luber, Johannes A Stork, Gian Diego Tipaldi, and Kai O Arras. People tracking with human motion predictions from social forces. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 464–469. IEEE, 2010.

[22] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *European Conference on Computer Vision*, pages 452–465. Springer, 2010.

[23] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[24] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen M. Rock. Randomized kinodynamic motion planning with moving obstacles. *I. J. Robotics Res.*, 21:233–256, 2002.

[25] Brandon Luders, Mangal Kothari, and Jonathan How. Chance constrained rrt for probabilistic robustness to environmental uncertainty. In *AIAA guidance, navigation, and control conference*, page 8160, 2010.

[26] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928–1935. IEEE, 2008.

[27] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. Generalized velocity obstacles. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5573–5578. IEEE, 2009.

[28] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, 2007.

[29] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[30] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

[31] Anirudh Vemula, Katharina Muelling, and Jean Oh. Path planning in dynamic environments with adaptive dimensionality. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[32] Kalin Gochev, Benjamin Cohen, Jonathan Butzke, Alla Safonova, and Maxim Likhachev. Path planning with adaptive dimensionality. In *Fourth annual symposium on combinatorial search*, 2011.

[33] David Silver. Cooperative pathfinding. In *AIIDE*, pages 117–122, 2005.

[34] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[35] Martin Rufli, Dave Ferguson, and Roland Siegwart. Smooth path planning in constrained environments. In *IEEE International Conference on Robotics and Automation, 2009.*, pages 3780–3785. IEEE, 2009.

[36] Jur Van Den Berg, Dave Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *IEEE International Conference on Robotics and Automation, 2006.*, pages 2366–2371. IEEE, 2006.

[37] Kostas E Bekris and Lydia E Kavraki. Greedy but safe replanning under kinodynamic constraints. In *IEEE International Conference on Robotics and Automation, 2007*, pages 704–710. IEEE, 2007.

[38] Stephane Petti and Thierry Fraichard. Safe motion planning in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005).*, pages 2210–2215. IEEE, 2005.

[39] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA), 2011*, pages 5628–5635. IEEE, 2011.

[40] Venkatraman Narayanan, Mike Phillips, and Maxim Likhachev. Anytime safe interval path planning for dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012*, pages 4708–4715. IEEE, 2012.

[41] Roland Philippsen and Roland Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *ICRA*, 2003.

[42] Ross A Knepper and Alonzo Kelly. High performance state lattice planning using heuristic look-up tables. In *IROS*, pages 3375–3380, 2006.

[43] Kalin Gochev, Alla Safonova, and Maxim Likhachev. Planning with adaptive dimensionality for mobile manipulation. In *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 2944–2951. IEEE, 2012.

[44] Kalin Gochev, Alla Safonova, and Maxim Likhachev. Incremental planning with adaptive dimensionality. In *ICAPS*, 2013.

[45] Haojie Zhang, Jonathan Butzke, and Maxim Likhachev. Combining global and local planning with guarantees on completeness. In *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 4500–4506. IEEE, 2012.

[46] Aleksandr Kushleyev and Maxim Likhachev. Time-bounded lattice for efficient planning in dynamic environments. In *IEEE International Conference on Robotics and Automation, 2009*, pages 1662–1668. IEEE, 2009.

[47] Janko Petereit, Thomas Emter, and Christian Walter Frey. Mobile robot motion planning in multi-resolution lattices with hybrid dimensionality. In *Proceedings of the IFAC Intelligent Autonomous Vehicles Symposium*, pages 546–563, 2013.

[48] Janko Petereit, Thomas Emter, and Christian W Frey. Combined trajectory generation and path planning for mobile robots using lattices with hybrid dimensionality. In *Robot Intelligence Technology and Applications 2*, pages 145–157. Springer, 2014.

[49] Anirudh Vemula, Katharina Mülling, and Jean Oh. Modeling cooperative navigation in dense human crowds. *CoRR*, abs/1705.06201, 2017. URL http://arxiv.org/abs/1705.06201.

[50] Simon Thompson, Takehiro Horiuchi, and Satoshi Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, pages 663–668. IEEE, 2009.

[51] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24 (1):31–48, 2005.

[52] F. Large, D. Vasquez, T. Fraichard, and C. Laugier. Avoiding cars and pedestrians using velocity obstacles and motion prediction. *IEEE Intelligent Vehicles Symposium, 2004*, pages 375–379, 2004.

[53] Edward T Hall. A system for the notation of proxemic behavior. *American anthropologist*, 65(5): 1003–1026, 1963.

[54] Dirk Helbing and Anders Johansson. *Pedestrian, crowd and evacuation dynamics*. Springer, 2011.

[55] Anders Johansson, Dirk Helbing, and Pradyumn K Shukla. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in complex systems*, 10(supp02):271–288, 2007.

[56] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1160–1168. ACM, 2006.

[57] Mikael Svenstrup, Thomas Bak, and Hans JAndersen. Trajectory planning for robots in dynamic human environments. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4293–4298, 2010.

[58] Ninad Pradhan, Timothy Burg, and Stan Birchfield. Robot crowd navigation using predictive position fields in the potential function framework. In *American Control Conference (ACC), 2011*, pages 4628–4633. IEEE, 2011.

[59] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *I. J. Robotics Res.*, 17:760–772, 1998.

[60] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*, 2012.

[61] Mark Pfeiffer, Ulrich Schwesinger, Hannes Sommer, Enric Galceran, and Roland Siegwart. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2096–2101, 2016.

[62] Kihwan Kim, Dongryeol Lee, and Irfan A. Essa. Gaussian process regression flow for analysis of motion trajectories. *2011 International Conference on Computer Vision*, pages 1164–1171, 2011.

[63] Carl E. Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning. In *Adaptive computation and machine learning*, 2009.

[64] Brian D. Ziebart, Nathan D. Ratliff, Garratt Gallagher, Christoph Mertz, Kevin M. Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha S. Srinivasa. Planning-based prediction for pedestrians. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936, 2009.