Robot Navigation in Dense Crowds: Statistical Models and Experimental Studies of Human Robot Cooperation

Thesis by

Pete Trautman

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2013

(Defended April 28, 2012)

To Jenna and Noah, Pat and Fred, and Mary and Rob.

Acknowledgements

My deepest gratitude goes to: Richard Murray, Andreas Krause, Joel Burdick, and Jim Beck. Each of you provided immense perspective on a vexing problem.

I would also like to thank the staff of Chandler dining hall at Caltech—in particular, Peter Daily and Jaime Reyes. Without your incredible enthusiasm for technology and robots I would not have been able to complete my experiments. If only all students of human-robot interaction were so lucky to have the support of people like you!

Abstract

This thesis explores the problem of mobile robot navigation in dense human crowds. We begin by considering a fundamental impediment to classical motion planning algorithms called the *freezing robot problem*: once the environment surpasses a certain level of complexity, the planner decides that all forward paths are unsafe, and the robot freezes in place (or performs unnecessary maneuvers) to avoid collisions. Since a feasible path typically exists, this behavior is suboptimal. Existing approaches have focused on reducing predictive uncertainty by employing higher fidelity individual dynamics models or heuristically limiting the individual predictive covariance to prevent overcautious navigation. We demonstrate that both the individual prediction and the individual predictive uncertainty have little to do with this undesirable navigation behavior. Additionally, we provide evidence that dynamic agents are able to navigate in dense crowds by engaging in joint collision avoidance, cooperatively making room to create feasible trajectories. We accordingly develop *interacting Gaussian processes*, a prediction density that captures cooperative collision avoidance, and a "multiple goal" extension that models the goal driven nature of human decision making. Navigation naturally emerges as a statistic of this distribution.

Most importantly, we empirically validate our models in the Chandler dining hall at Caltech during peak hours, and in the process, carry out the first extensive quantitative study of robot navigation in dense human crowds (collecting data on 488 runs). The multiple goal interacting Gaussian processes algorithm performs comparably with human teleoperators in crowd densities nearing 1 person/m^2, while a state of the art noncooperative planner exhibits unsafe behavior more than 3 times as often as the multiple goal extension, and twice as often as the basic interacting Gaussian process approach. Furthermore, a reactive planner based on the widely used *dynamic window* approach proves insufficient for crowd densities above 0.55 people/m^2. We also show that our noncooperative planner or our reactive planner capture the salient characteristics of nearly any dynamic navigation algorithm. For inclusive validation purposes, we show that either our non-interacting planner or our reactive planner captures the salient characteristics of nearly any existing dynamic navigation algorithm. Based on these experimental results and theoretical observations, we conclude that a cooperation model is critical for safe and efficient robot navigation in dense human crowds.

Finally, we produce a large database of ground truth pedestrian crowd data. We make this ground truth database publicly available for further scientific study of crowd prediction models, learning from demonstration algorithms, and human robot interaction models in general.

Contents

Chapter 1

Introduction

In this chapter, we review the existing literature on robot navigation in human crowds, introduce the freezing robot problem, and provide a conceptual explanation of why modeling a cooperative interaction between humans and robots is critical for successful crowd navigation. We finish the chapter by detailing the contributions and the organization of the thesis.

1.1 Motivation

One of the first major deployments of an autonomous robot in an unscripted human environment occurred in the late 1990s at the Deutsches Museum in Bonn, Germany (Burgard et al. [19]). This RHINO experiment was quickly followed by another robotic tour guide experiment; the robot in the follow-on study, named MINERVA (Thrun et al. [109]), was exhibited at the Smithsonian and at the National Museum of American History in Washington D.C. Both the RHINO and MINERVA robots made extensive use of probabilistic methods for localization and mapping (Roy et al. [90], Dellaert et al. [24], Roy and Thrun [89]). Additionally, these experiments pioneered the nascent field of human robot interaction in natural spaces—see Schulte et al. [94] and Thrun et al. [110]. Perhaps most importantly, the RHINO and MINERVA studies inspired a wide variety of research in the broad area of robotic navigation in the presence of humans, ranging from additional work with robotic tour guides (Shiomi et al. [98], Siegwart et al. [101], Shiomi et al. [99], Eppstein et al. [31], Foka and Trahanias [34], Bauer et al. [8] and Hayashi et al. [45]), to work on nursing home robots (Pineau et al. [81], Montemerlo et al. [76] and Roy et al. [91]), to robots that perform household chores (Srinivasa et al. [103] and Kruse et al. [62]), to field trials for interacting robots as social partners (Kanda et al. [54], Saiki et al. [93], Kruse and et al. [61] and Seifer and Matarić [96]), to decorum for robot hosts (Sidner and Lee [100] and Kanda et al. [55]), and even to protocols for social robot design (Glas et al. [41]).

Despite the many successes of the pioneering RHINO and MINERVA experiments, and the success of the work that followed it, fundamental questions about robotic navigation in dense human

crowds remain unresolved. In particular, prevailing algorithms for navigation in dynamic environments emphasize deterministic and decoupled prediction algorithms (such as in LaValle [68], Latombe [66] and Choset et al. [22]), and are thus inappropriate for applications in highly uncertain environments or for situations in which the agent and the robot are dependent on one another. Critically, a large-scale experimental study of robotic navigation in dense human crowds is unavailable.

In this thesis, we focus on these two deficiencies: a dearth of human-robot cooperative navigation models and the absence of a systematic study of robot navigation in dense human crowds. We thus develop a cooperative navigation methodology and conduct the first extensive ($n_{\text{runs}} \approx 500$) field trial of robot navigation in *natural* human crowds.

1.2 Related Work

Independent agent constant velocity Kalman filters are a starting point for modeling the uncertainty in dynamic environments. Unfortunately, this prediction engine can lead to an uncertainty explosion that makes safe and efficient navigation impossible (Figure 1.1). Some research has thus focused on *controlling* this predictive uncertainty. For instance, in Thompson et al. [108], Bennewitz et al. [11], Helble and Cameron [49] and Large et al. [65], high fidelity independent human motion models were developed, in the hope that controlling the predictive uncertainty would lead to improved navigation performance. The work in Du Toit and Burdick [30] and Du Toit [28] improves navigation performance by directly limiting individual agent predictive uncertainty. Specifically, they formalize robot motion planning in dynamic, uncertain environments as a stochastic dynamic program (see Bertsekas [12, 13]); intractability is avoided with receding horizon control techniques (Mayne et al. [71], Morari and Lee [77] and Carson [20]). Furthermore, the collision checking algorithms developed in earlier work (see Du Toit and Burdick [29], which has its roots in Blackmore [15], Blackmore et al. [17] and Blackmore and Williams [16]) keeps the navigation protocol safe. The insight is that since replanning is used, the predictive covariance can be held constant at measurement noise. Although robot-agent interaction models are developed for a few cases, the primary contribution from this line of research comes in the form of independent agent dynamics models. Section 2.4 argues that only limiting the uncertainty explosion is insufficient for robot navigation in dense crowds.

The work of Aoude et al. [5, 4] and Joseph et al. [52] shares insight with the approach of Du Toit [28], although more sophisticated individual models are developed: motion patterns are modeled as a Gaussian process mixture (Rasmussen and Williams [84]) with a Dirichlet Process prior over mixture weights (Teh [107]). The Dirichlet process prior allows for representation of an unknown number of motion patterns, while the Gaussian process allows for variability within a particular motion pattern. Rapidly exploring random trees (RRT, see LaValle and Kuffner [67]) are used to find feasible paths. Similar emphasis is placed on probabilistic collision checking by incorporating

the earlier work in Aoude et al. [3]. No work is done on modeling agent interaction.

The field of proxemics (Hall [42, 43]) has much to say about the interaction between a navigating robot and a human crowd. Specifically, proxemics tries to understand human proximity relationships, and in so doing, can provide insight about the design of social robots. In Mead et al. [73, 74] and Takayama and Pantofaru [105] various robots are developed in accordance with proxemic rules, while in Mead and Matarić [72] a probabilistic framework for identifying specific proxemic indicators is developed. Similarly, Castro-Gonzalez et al. [21] studies pedestrian crossing behaviors using proxemics. However, this work only studies sparse crowd interactions in scripted settings.

In Svenstrup et al. [104] rapidly-exploring random trees are combined with a potential field (Khatib [56], Koren and Borenstein [60]); the values in this potential field are based on proxemics. The authors of Pradhan et al. [83] take a similar proxemic potential function based approach. Although these navigation algorithms model human-robot interaction, they do not model human-robot *cooperation*. Instead, the emphasis is placed on respecting a proper distance between the robot and the humans (similar to the work of Ziebart et al. [120]). Further, the algorithm is implemented in simulation only, and the density of humans in the simulated robotic workspace is kept quite low (approximately 0.1 person/m^2).

Rios-Martinez et al. [87] take a "human-centric" approach as well, but instead of using the proxemic rules of Hall [42], they use the criteria of Lam et al. [64] instead. They incorporate these rules of personal space into the robot's behavior by extending the Risk-RRT algorithm developed in Fulgenzi et al. [40]. The Risk-RRT algorithm extends the traditional RRT algorithm to include risk, or the probability of collision along any candidate trajectory.

The mobile robot navigation research in Althoff et al. [1] is more agnostic about the specific cultural considerations of the dynamic agents. A "probabilistic collision cost" is introduced (to assess the fitness of candidate robot trajectories in human crowds) that is based on the idea of *inevitable collision states*, described in Fraichard and Asama [37] and expanded in Bautin et al. [10] (inevitable collision states are robot configurations that are guaranteed to result in a collision with another agent). In particular, Fraichard [36] advocates three quantities as essential to the proper evaluation of motion safety: the dynamics of the robot, the dynamics of the environment, and a long enough time horizon. Furthermore, in Fraichard [36] it is argued that full knowledge of these quantities would enable perfect prediction, which in turn would guarantee perfect collision avoidance. The cost function of Althoff et al. [1] encodes an approximation of these rules. Importantly, collision avoidance capabilities of neighboring dynamic agents are modeled. However, experiments are carried out entirely in simulation.

Importantly, work has been done on learning navigation strategies by observing many example trajectories. In Ziebart et al. [120], a combination of inverse reinforcement learning and the principle of maximum entropy is used to learn taxi cab driver decision making protocols from large volumes of

data. These methods are extended to the case of a robot navigating through an office environment in Ziebart et al. [119]: pedestrian decision making is first learned from a large trajectory example database, and then the robot navigates in a way that causes the least disruption to the human's *anticipated* paths. In Henry et al. [50], the authors extend inverse reinforcement learning to work in dynamic environments. Their planner is trained using *simulated* trajectories, and the method recovers a planner which duplicates the behavior of the simulator. In the work of Waugh et al. [118], agents *learn* how to act in multi-agent settings using game theory and the principle of maximum entropy. The work of Kuderer et al. [63] leverages IRL to learn an interaction model from human trajectory data. Critically, the IRL feature vector is an extension of the cooperation model that was developed in Trautman and Krause [114]; thus, not only does this work model cooperation, it pioneers IRL navigation strategies from *real* human interaction data as well. However the experiments are limited in scope—one scripted human crosses paths with a single robot in a laboratory environment.

We mention briefly that (although not developed in the field of robotic navigation) models capturing crowd interaction are explored in Pellegrini et al. [79, 80] and Luber et al. [70] for the purposes of crowd prediction. These papers rely on *the social forces model*, developed in Helbing and Molnar [46] and Helbing et al. [48, 47]. The ideas introduced in Helbing and Molnar [46] underpin the interaction model of Chapter 3.

We thus suggest that there is a dearth of human-robot cooperative navigation models, and no extensive study of robot navigation in dense human crowds has taken place. In this thesis, we address these two deficiencies.

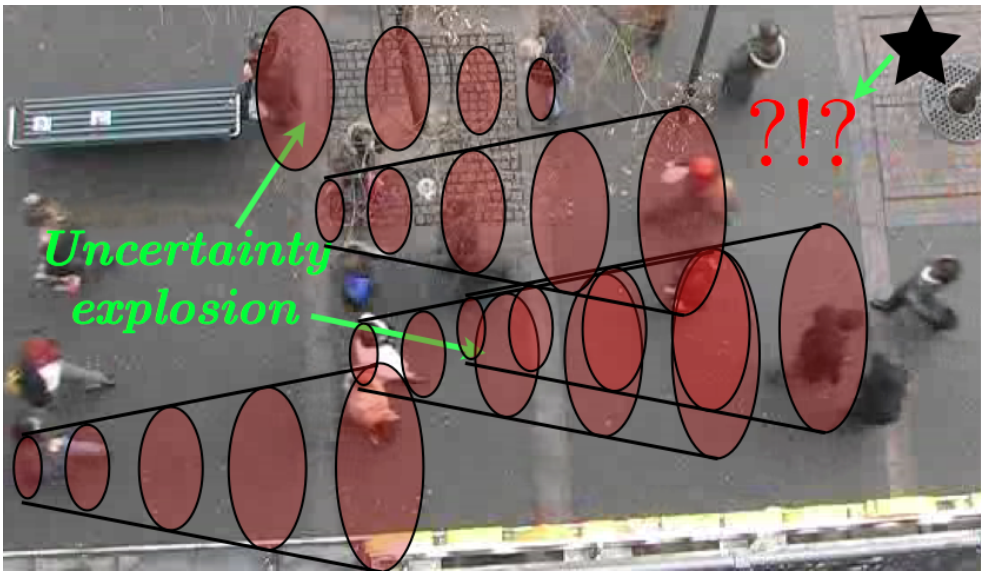


Figure 1.1: Freezing robot problem as a result of unconstrained prediction. The robot is represented as the black star, and the ellipses represent the predictive covariance. The question marks indicate that the robot can find no clear path forward.

1.3 The Freezing Robot Problem

In Figure 1.1, we illustrate the *freezing robot problem*. The black star (representing a mobile robot) predicts the individual trajectories (light red ellipses) of a crowd of people. In this case, the lack of *any* predictive covariance constraints results in a robot that cannot make an informed navigation decision: the deficiencies of the predictive models force the robot to come to a complete stop (or the robot chooses to follow an essentially arbitrary path through the crowd). As we discuss in Section 5.5.1.1 , arbitrary and highly evasive paths can often be much worse than suboptimal—they can be dangerous.

Figure 1.1 suggests that the culprit behind the freezing robot problem could be the *individual* uncertainty explosion. Indeed, if the *amount* of uncertainty was the primary reason for this suboptimal navigation, then using more precise individual dynamics models would prevent the freezing robot problem. As is illustrated in Figure 1.2, this approach works well for certain crowd configurations.

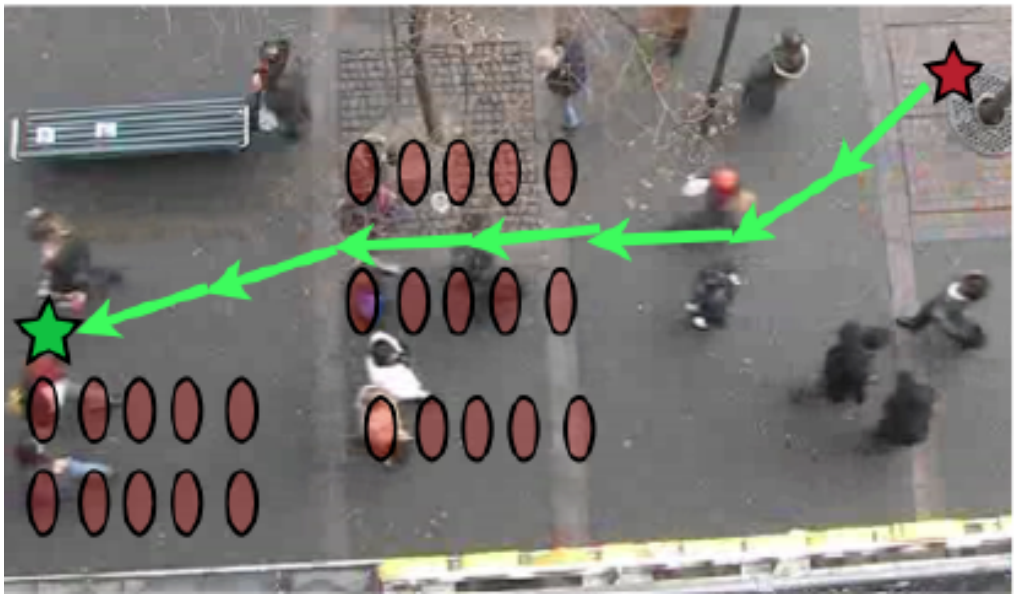


Figure 1.2: If the predictive covariance of individual agents is held to a small value, navigation can proceed in an optimal manner—if the crowd is sparse enough.

However, in Section 2.4.1, we show that even under *perfect individual prediction* (i.e., each agent's trajectory is *known* to the planning algorithm) the freezing robot problem still occurs if the crowd adopts specific configurations. In Figures 1.3(a) and 1.3(b) we illustrate a very common crowd configuration that can cause any independent planner to fail; when people walk shoulder to shoulder, the robot is forced to walk *around* the crowd, even when the humans are willing to allow passage. In more demanding scenarios, like the cafeteria illustration in Figure 1.3(c), this behavior can lead to a failure mode—for instance, the robot in this run collided with the wall in an attempt to make

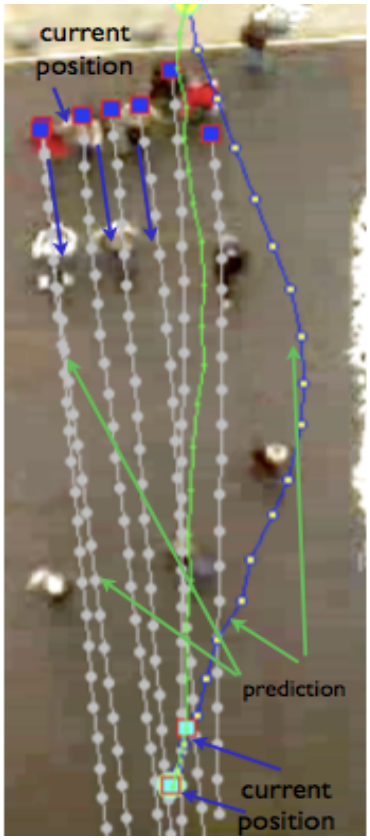
way for the humans.

As was highlighted in Section 1.2, existing robot navigation approaches commonly ignore mathematical models of cooperation between humans and robots. Unfortunately, under this modeling assumption, the freezing robot problem will always occur, given dense enough crowds.

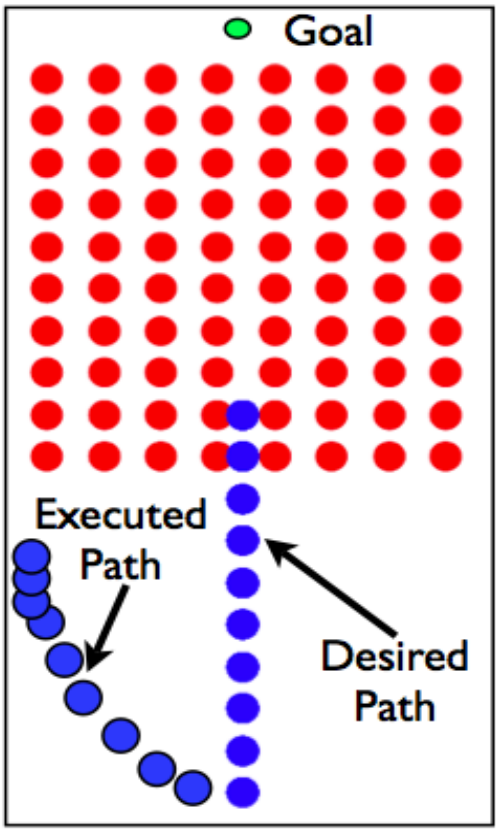
Given this observation, how is it possible that people can safely navigate through crowds? The key insight is that people typically engage in *joint* collision avoidance (this is similar to the *social forces model* of Helbing and Molnar [46] and Helbing et al. [48, 47]): they adapt their trajectories to each other to make room for navigation (see Figure 1.4).

Evidence of the usefulness of joint collision avoidance models occurs in other fields as well: work on multi-robot coordination in van den Berg et al. [115, 116, 117] and Snape et al. [102] shows that robots programmed to jointly avoid each other are guaranteed to be collision free and display vastly improved efficiency at navigation tasks. Additionally, this joint collision avoidance criteria has been used to improve the data association and target tracking of individuals in human crowds (Pellegrini et al. [79, 80], Luber et al. [70]).

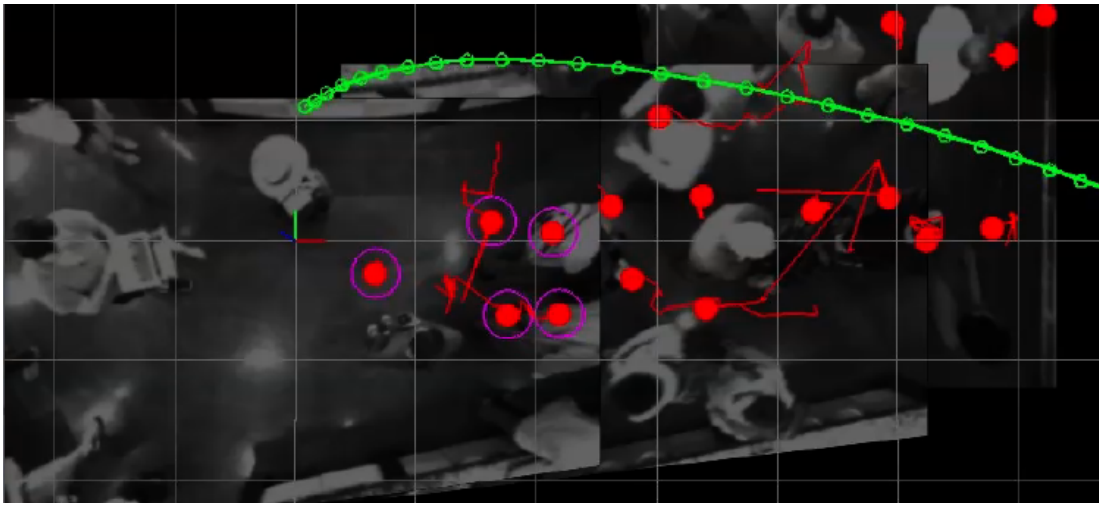
To our knowledge, however, this principle has not been used to improve *robot* navigation in *human* crowds. Thus, the central idea of this thesis is to explicitly model human-robot interaction and cooperation in crowds (illustrated in Figure 1.4). To this end, we develop *interacting Gaussian processes*, a principled statistical model, based on dependent output Gaussian Processes. IGP describes a probabilistic interaction between multiple navigating entities.



(a)



(b)



(c)

Figure 1.3: **(a)** Even if we hold pedestrian predictive covariance to be extremely small (grey circles), common crowd configurations (shoulder to shoulder walking, sparse crowd) can lead to evasive maneuvering by the robot **(b)** An illustration of what is occurring in panel **(a)**. Red dots represent crowd prediction, blue dots represent robot decision making **(c)** Example of freezing robot problem in cafeteria. Robot was not anticipating interaction, and so chose a highly evasive maneuver (green line). Inspection of human tracks (red lines), in contrast, show people passing between each other. *Imagine a crowd of agents unaware of joint interaction—that is, imagine a room full of agents all trying to move along the wall.*

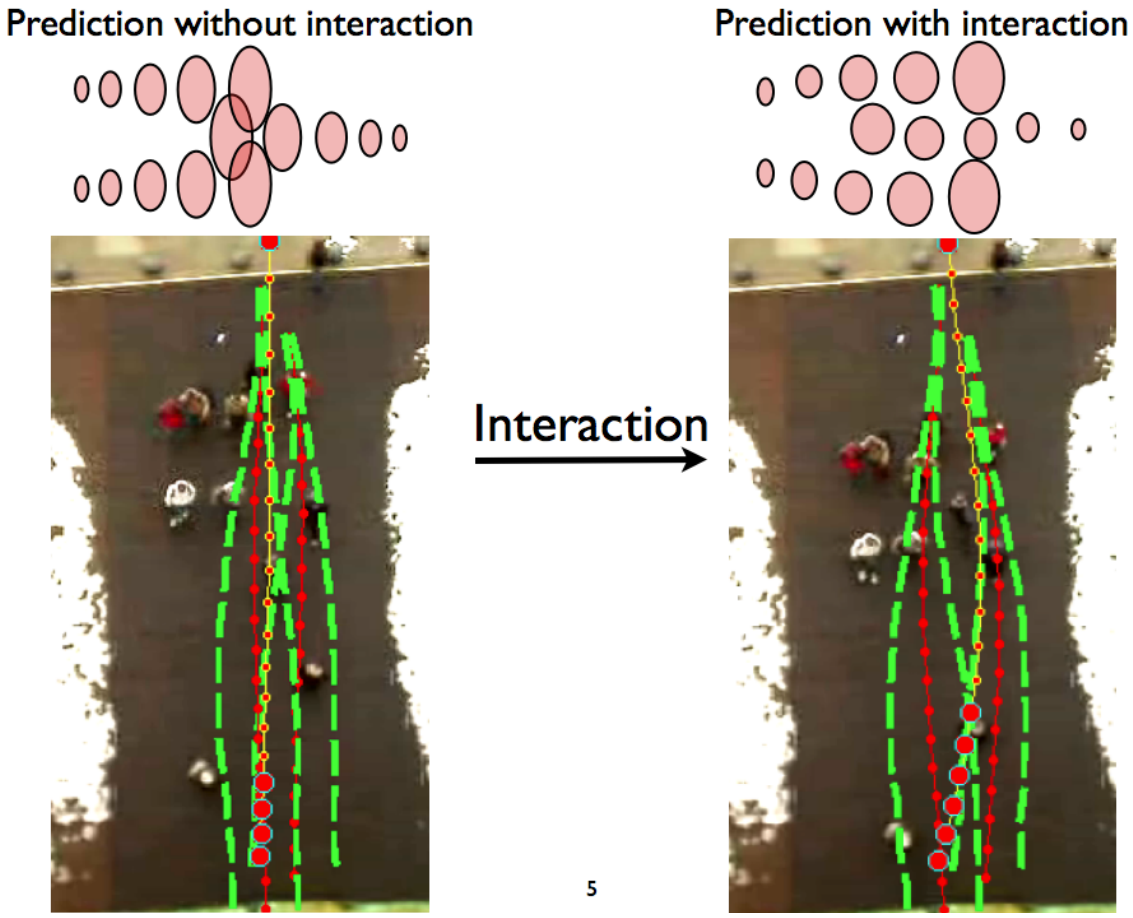


Figure 1.4: Including interaction in the predictive models causes the prediction tubes to bend around one another, in a *jointly* cooperative way. As this thesis will show, anticipating cooperation is a prerequisite for successful navigation in dense human crowds.

1.4 Thesis Contributions and Organization

The remainder of this thesis is organized as follows. Chapter 2 provides further technical background for the thesis. We define and identify the "freezing robot problem" as a critical impediment to robot navigation in dense human crowds, and introduce various approaches for solving the freezing robot problem. Additionally, Chapter 2 introduces Gaussian processes as a novel modeling extension to existing Markov prediction techniques (such as the Kalman filter).

In Chapter 3, a novel method for modeling cooperation is described, called *interacting Gaussian Processes*. In particular, this prediction density captures cooperative collision avoidance, the non-Markov nature of agent trajectories, and the goal driven nature of human decision making. Additionally, by formulating navigation in dense crowds as a *density estimation* problem, rather than a cost optimization problem, we discover an equivalence between inference and planning. This equivalence provides a novel solution methodology since we can exploit existing approximate inference methods to determine navigation strategies. In particular, we use importance sampling to approximate the interacting Gaussian processes density. The chapter finishes with a simulation of how modeling cooperation can improve navigation performance. The results of this simulation serves as motivation for the quantitative study detailed in Chapter 5.

Chapter 4 provides details about the construction of the robot navigation experiment. We provide a description of the robotic workspace in Chandler dining hall at Caltech, the pedestrian tracking system, and the robot itself.

In Chapter 5, we provide the first extensive quantitative study of robot navigation in dense human crowds (488 runs completed), specifically testing how cooperation models effect navigation performance. Importantly, we validate the navigation models developed in this thesis, finding, in particular, that the multiple goal interacting Gaussian processes algorithm performs comparably with human teleoperators in crowd densities near 1 person/m^2, while a state of the art noncooperative planner exhibits unsafe behavior more than 3 times as often as this multiple goal extension, and more than twice as often as the basic interacting Gaussian processes. Furthermore, we find that a reactive planner based on the widely used "dynamic window" approach fails for crowd densities above 0.55 people/m^2. We also show that our noncooperative planner or our reactive planner capture the salient characteristics of nearly any dynamic navigation algorithm. Based on these experimental results and theoretical observations, we conclude that a cooperation model is critical for safe and efficient robot navigation in dense human crowds.

Finally, Chapter 6 summarizes the contributions of this thesis, provides details on future work opportunities, and potential applications areas for this technology.

Chapter 2

Background and Problem Setup

This chapter provides context for the contributions of this thesis. First, since this research is primarily concerned with robot navigation, we describe some common decision making frameworks. We introduce robot path planning, decision making under uncertainty and in dynamic environments, and the principle of receding horizon control. We also describe some approaches to prediction of dynamic entities. The freezing robot problem, which we discuss in the final section of this chapter, serves as the motivation for the methods developed in this thesis.

2.1 Discrete Stochastic Optimal Control

We begin by defining some quantities from the field of discrete optimal planning. Let $f_t \in \mathcal{F} \subseteq \mathbb{R}^{n_f}$ be an element of the state space $\mathcal{F}$. The control (or action) is denoted by $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, where $\mathcal{U}$ is the action space. Additionally, let ω_k (the uncertainty in the system) be distributed according to some distribution $p(\omega_k)$. We define the transition function $\beta \colon \mathcal{F} \times \mathcal{U} \times \mathcal{W} \to \mathcal{F}$ between temporal states as

$$f_t = \beta(f_{t-1}, u_{t-1}, w_{t-1}).$$

Equivalently, we can specify the *transition probability density*

$$p(f_t \mid f_{t-1}, u_{t-1}),$$

where the uncertainty introduced by ω_{t-1} produces a probability density function over the next state. We remark that by applying marginalization, the chain rule of probability (Appendix A.1), and with knowledge of the initial distribution of the state $p(f_1)$, we can recover

$$p(f_2) = \int p(f_2 \mid f_1) p(f_1) df_1.$$

By iteration, then, we can recover the distribution over any state

$$p(f_T) = \int p(f_T \mid f_{T-1}) p(f_{T-1}) df_{T-1},$$

where T is the end of our prediction horizon. We point out that an implicit dependency assumption is being made here: each temporal state f_t is only dependent on the most recent temporal state f_{t-1}; this is because the transition function is assumed to be *first order Markov*. In Chapter 3 we generalize the temporal dependency to arbitrary lengths using Gaussian processes for our trajectory models. Correspondingly, we introduce the boldface notation

$$\mathbf{f} = (f_1, f_2, \dots f_T)$$

as a shorthand for the entire trajectory. We will use boldface to indicate any quantity over the duration of an entire trajectory.

Consider further *control policies* $\pi \colon \mathcal{F} \to \mathcal{U}$ that map states to controls

$$u_t = \pi_t(f_t).$$

Additionally, we define the sequence of control policies over T steps to be $\boldsymbol{\pi} = \{\pi_1, \pi_1, \dots, \pi_T\}$, and we define a cost function in terms of the control policy:

$$c(\boldsymbol{\pi}) = l_{T+1}(f_{T+1}) + \sum_{t=1}^{T} l_t(f_t, \pi_t),$$

where $l_t(f_t, \pi_t)$ is the stage additive cost function. Thus, the expected (Appendix A.3) cost of policy $\boldsymbol{\pi}$ is

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\mathbf{f}} \left[c(\boldsymbol{\pi}) \right].$$

The optimal policy, $\boldsymbol{\pi}^* = \{\pi_0^*, \pi_1^*, \dots, \pi_{T-1}^*\}$, minimizes the expected cost over the set of all admissible policies $\tilde{\boldsymbol{\pi}}$:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi} \in \tilde{\boldsymbol{\pi}}} J(\boldsymbol{\pi})$$

with the optimal cost given by

$$J(\boldsymbol{\pi}^*) = \mathbb{E}\left[l_{T+1}(f_{T+1}) + \sum_{t=1}^{T} l_t(f_t, \pi_t^*) \right]$$

Finding the optimal control policy is typically non-trivial; a common approach is to use *Dynamic*

Programming. For details on dynamic programming, please consult Bertsekas [12].

For simplicity, we presented here the case of policy search with *perfect state information*—that is, we assumed that we could measure the state f_t *directly* through the present time t. This is a strong assumption, since typical applications involve sensors that produce noisy measurements of the state, rather than the state itself. Fortunately, if we generalize state space to *belief space*, the formulation above (including the use of dynamic programming to find solutions) can still be used. For details on this transformation consult LaValle [68].

2.2 Receding Horizon Control

The framework of receding horizon control computes policies through online solution of a finite time optimal planning problem (as in Section 2.1) that can enforce explicit state and control constraints. In receding horizon control, the computed plan is applied to the system in an open loop manner over some time interval shorter than the planning horizon. As new information arrives, a new optimal policy is found and executed. Recursively planning in this manner provides a type of closed-loop feedback by incorporating current state information into the plan currently being executed.

Mathematically, receding horizon control finds the policy at time t'

$$\boldsymbol{\pi}^*_{t'} = \{\pi_{t'}, \pi_{t'+1}, \ldots, \pi_T\}$$

that optimizes

$$J(\boldsymbol{\pi}_{t'}) = \mathbb{E}\left[c(\boldsymbol{\pi}_{t'})\right]$$

When implemented in this manner, receding horizon control has proven to be a powerful approximate planning method (see Carson [20] and Du Toit [28] for example applications).

2.3 Dynamic Agent Prediction

In this section, we present methods for dynamic agent prediction. Kalman filter based prediction chooses a linear dynamics model for the prediction step (see Appendix A.5), while ignoring the correction (or update) step, all in a first order Markov framework. Gaussian processes for independent trajectory models can be viewed as an extension of the Kalman filtering framework.

2.3.1 Kalman Filter Based Prediction

Kalman filters (Kalman [53]) are a special case solution of the sequential Bayesian estimation equations presented in Appendix A.5. We use $\mathbf{x}_t \in \mathbb{R}^{n_x}$ to denote the *state* variable and $\mathbf{z}_t \in \mathbb{R}^{n_z}$

to denote the measurement variable at time t. Furthermore, for the Kalman filter recursion to be applicable, the dynamics model $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$, the measurement likelihood $p(\mathbf{z}_t \mid \mathbf{x}_t)$, and the distribution encoding initial knowledge $p(\mathbf{x}_0)$ all must be Gaussian. These requirements ensure that the posterior $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$ is also Gaussian. Rather than specifying these distributions directly, however, the system and measurement equations can be described:

$$\begin{aligned}\mathbf{x}_t &= A_{t-1}\mathbf{x}_{t-1} + B_{t-1}u_{t-1} + F_{t-1}\omega_{t-1}\\ \mathbf{z}_t &= C_t\mathbf{x}_t + H_t\nu_t.\end{aligned}$$

Here, $\omega_t \sim \mathcal{N}(0, W)$ and $\nu_t \sim \mathcal{N}(0, V)$ are independent Gaussian noise terms with covariances W and V (where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$). The variable u_{t-1} is the control input, while the matrix A_{t-1} is the dynamics model, B_{t-1} is the control input model, F_{t-1} is the dynamics noise model, C_t is the measurement model, and H_t is the dynamics noise model.

This set of linear equations is equivalent to specifying the distributions themselves. For an interesting take on the system versus distribution point of view, see Ko and Fox [57], where the authors learn $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$ and $p(\mathbf{z}_t \mid \mathbf{x}_t)$ (for a remote controlled micro-blimp) using machine learning techniques, and compare the results to physics based descriptions of the matrices A_t, B_t and C_t.

2.3.1.1 Prior Kalman Filter

Before the measurement arrives, the Kalman Filter is integrated forward according to the prediction step of Appendix A.5

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t}) = \int p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)p(\mathbf{x}_t \mid \mathbf{z}_{1:t})d\mathbf{x}_t.$$

Since the dynamics models are linear and Gaussian, we have that

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t}) = \mathcal{N}\left(\mathbf{x}_{t+1} \mid \hat{\mathbf{x}}_{t+1:t}, \boldsymbol{\Sigma}_{t+1:t}\right),$$

where $\boldsymbol{\Sigma}_{t+1:t} \in \mathbb{R}^{n_x \times n_x}$. Thus, we need only predict forward the mean and the covariance:

$$\begin{aligned}\hat{\mathbf{x}}_{t+1:t} &= A_t\hat{\mathbf{x}}_{t:t} + B_t u_t\\ \boldsymbol{\Sigma}_{t+1:t} &= A_t\boldsymbol{\Sigma}_{t:t}A_t^\top + F_t W F_t^\top.\end{aligned}$$

An example model choice is to encode the zeroeth and first order derivatives in $\mathbf{x}_t$, let A_t be a constant velocity model, ignore inputs u_t (since the inputs of a dynamic agent are typically unknown), and capture random accelerations with the noise term $\omega_t \sim \mathcal{N}(0, W)$.

2.3.1.2 Posterior Kalman Filter

Once measurements are collected, the mean and covariance are corrected according to the update step of Appendix A.5

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t+1}) = \frac{p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t})}{p(\mathbf{z}_{t+1}|\mathbf{z}_{1:t})}.$$

Since the measurement and prediction models are linear and Gaussian, we have that

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t+1}) = \mathcal{N}\left(\mathbf{x}_{t+1} \mid \hat{\mathbf{x}}_{t+1:t+1}, \mathbf{\Sigma}_{t+1:t+1}\right)$$

and so we need only correct the mean and the covariance

$$\begin{aligned}
\hat{\mathbf{x}}_{t+1:t+1} &= \hat{\mathbf{x}}_{t+1:t} + K_{t+1}\left(\mathbf{z}_{t+1} - \hat{\mathbf{z}}_{t+1:t}\right) \\
\mathbf{\Sigma}_{t+1:t+1} &= \left(\mathbf{I} - K_{t+1}C_{t+1}\right)\Sigma_{t+1:t},
\end{aligned}$$

where

$$\begin{aligned}
\hat{\mathbf{z}}_{t+1:t} &= C_{t+1}\hat{\mathbf{x}}_{t+1:t} \\
K_{t+1} &= \Sigma_{t+1:t}C_{t+1}^{\top}\Gamma_{t+1:t}^{-1} \\
\Gamma_{t+1:t} &= C_{t+1}\mathbf{\Sigma}_{t+1:t}C_{t+1}^{\top} + H_{t+1}VH_{t+1}^{\top}.
\end{aligned}$$

We remark that in the absence of corrective measurements (as is the case for *prediction*), the covariance grows at each incrementing step as

$$\begin{aligned}
\mathbf{\Sigma}_{t+1:t+1} &= \left(\mathbf{I} - K_{t+1}C_{t+1}\right)\mathbf{\Sigma}_{t+1:t} \\
&= A_t\mathbf{\Sigma}_{t:t}A_t^{\top} + F_tWF_t^{\top} \\
&= \mathbf{\Sigma}_{t+1:t}.
\end{aligned}$$

Thus, without high fidelity models (i.e., small values for W), prediction can become uninformative very quickly. We discuss the consequences of this uncertainty explosion in Section 1.3 and in Chapter 2.4.

2.3.2 Gaussian Process Based Prediction

A Gaussian process (see Rasmussen and Williams [84], Ko and Fox [57, 58] and Li et al. [69]) is a distribution over (typically smooth) functions, and thus well-suited to model wheeled mobile robot trajectories. Formally, a Gaussian process is a collection of Gaussian random variables indexed by a set—in our case, the continuum of time steps $[1, T]$—that is parameterized uniquely by a mean

function

$$m\colon [1,T] \to \mathbb{R}$$

(typically taken as zero without loss of generality) and a covariance (or kernel) function

$$k\colon [1,T] \times [1,T] \to \mathbb{R}.$$

We will write

$$\mathbf{f}^{(i)} \sim GP(m^{(i)}, k^{(i)})$$

to mean that the *random function* $\mathbf{f}^{(i)}\colon [1,T] \to \mathbb{R}$ is distributed as a Gaussian process with mean $m^{(i)}$ and covariance $k^{(i)}$; since we will be generalizing to the case of multiple dynamic agents $i = 1, \dots n$, we introduce the superscript notation i to indicate a particular agent i. For clarification, we draw a comparison: with a Gaussian vector $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the matrix element $\boldsymbol{\Sigma}_{l,j}$ encodes the covariance between the elements of the state vector $\mathbf{x}_l$ and $\mathbf{x}_j$. Likewise, with Gaussian processes, the kernel function parameterizes the smoothness of the function: recalling that points $t \in [1,T]$ act as our index set, we see that $\mathbf{f}^{(i)}(t)$ and $\mathbf{f}^{(i)}(t')$ are related according to the value of $k^{(i)}(t,t')$.

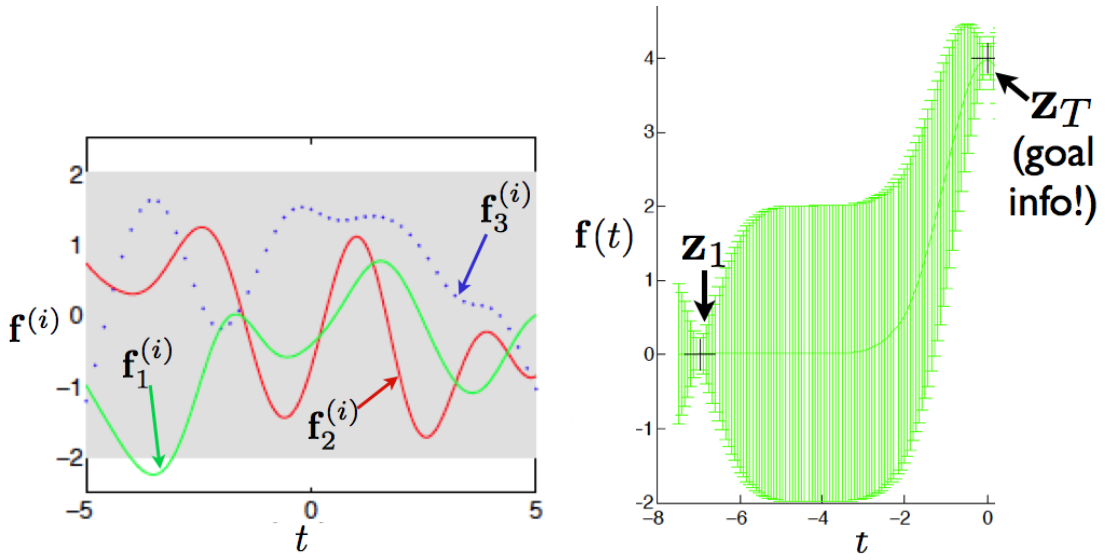
Notionally, we believe the *true* trajectory $\mathbf{f}_*^{(i)}$ exists (or will exist, since we have only gathered prior data about this trajectory; see Section 2.3.2.2). The Gaussian process $GP(m^{(i)}, k^{(i)})$ encodes all our prior knowledge about the function $\mathbf{f}_*^{(i)}$. In contrast, for sequential Bayesian estimation, the prior model is typically derived from first principles (such as the physics of the moving object), and encoded as the distribution $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$. With Gaussian processes, the prior model $GP(m^{(i)}, k^{(i)})$ is *learned* from training data. The dearth of high fidelity first principles models of human behavior, combined with the abundance of example human trajectory data, make Gaussian processes especially appealing for our application.

2.3.2.1 Posterior Gaussian Process

For simplicity of notation, we formalize our Gaussian process trajectory model for one-dimensional locations only. Multiple dimensions are easily incorporated by modeling each dimension as a separate Gaussian process.

Suppose that we collect the set of noisy measurements $\mathbf{z}_{1:t}^{(i)} = (\mathbf{z}_1^{(i)}, \dots, \mathbf{z}_t^{(i)})$ of the trajectory, where

$$\mathbf{z}_{t'}^{(i)} = \mathbf{f}^{(i)}(t') + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma_{noise}^2).$$



(a) (b)

Figure 2.1: **(a)** Possible trajectory samples $\{\mathbf{f}_k^{(i)}\}_{k=1}^3 \sim p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)})$ from a particular agent i **(b)** Using goal information $\mathbf{z}_T$ to constrain trajectory prediction.

Then we can calculate the posterior Gaussian process $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) = GP(m_t^{(i)}, k_t^{(i)})$, where

$$m_t^{(i)}(t') = \Sigma_{1:t,t'}^{\mathrm{T}}(\Sigma_{1:t,1:t} + \sigma_{noise}^2\mathbf{I})^{-1}\mathbf{z}_{1:t}^{(i)}$$
$$k_t^{(i)}(t_1, t_2) = k^{(i)}(t_1, t_2) - \Sigma_{1:t,t_1}^{\mathrm{T}}(\Sigma_{1:t,1:t} + \sigma_{noise}^2\mathbf{I})^{-1}\Sigma_{1:t,t_2}.$$

Hereby, $\Sigma_{1:t,t'} = [k^{(i)}(1,t'), k^{(i)}(2,t'), \ldots, k^{(i)}(t,t')]$, and $\Sigma_{1:t,1:t}$ is the matrix such that the (l,j) entry is $\Sigma_{l,j} = k^{(i)}(l,j)$ and the indices (l,j) take values from $1:t$. The quantity σ_{noise}^2 is the measurement noise (which is assumed to be Gaussian, and as shown in Section 2.3.2.3, can be learned from training data). Since the entire trajectory $\mathbf{f}^{(i)}\colon [1,T] \to \mathbb{R}$ is being modeled, information about the goal of the agent (such as an eating station in a cafeteria) can be treated as a measurement $\mathbf{z}_T^{(i)}$. As illustrated in Figure 2.1(b), the information $\mathbf{z}_T^{(i)}$ constrains the predictive uncertainty along the entirety of the trajectory $\mathbf{f}^{(i)}$ (not only at time T).

2.3.2.2 Training the Gaussian Process

The kernel function $k^{(i)}$ is the crucial ingredient in a Gaussian process model, since it encodes "how" the underlying function behaves: in our case, how a dynamic agent moves (e.g., how smoothly, how linearly, length scales of behavior modes, etc). For a kernel function to be valid it must first be positive semidefinite. That is, for all sets A that take values in the indexing set (for our case, the indexing set is the closed continuum $[1,T]$), $\Sigma_{A,A}$ must be positive definite. A class of useful kernel functions are known and are discussed in detail in Rasmussen and Williams [84]. These individual

kernels can be combined to make new kernels via summation and multiplication.

However, even with this set of predefined kernel functions and rules for combining them, choices still have to be made. What combination of discrete kernel functions should be used for a particular application? And once we decide on the kernel functions, how should the kernel hyperparameters be chosen?

To answer these questions, we begin by assuming that we are presented with a *training set* of input-output pairs. For our pedestrian dynamics models, the inputs are the times $t' = 1, 2, \ldots, t$ and the outputs are the trajectory measurements $\mathbf{z}_{1:t}^{(i)}$. Using this training data we can optimize over both specific kernel functions as well as the hyperparameters of those particular kernel functions. Additionally, we describe how *a priori* information can be leveraged to inform our choice of kernel function.

Gaussian Process Marginal Likelihood In order to optimize our kernel according to the training data set, we first calculate the probability of the data given the hyperparameters $\boldsymbol{\theta}$ and input times $\{1, \ldots, t\}$:

$$p(\mathbf{z}_{1:t}^{(i)} \mid \{1, \ldots, t\}, \boldsymbol{\theta}) = \int p(\mathbf{z}_{1:t}^{(i)} \mid \mathbf{f}^{(i)}, \{1, \ldots, t\}, \boldsymbol{\theta}) p(\mathbf{f}^{(i)} \mid \{1, \ldots, t\}, \boldsymbol{\theta}) d\mathbf{f}^{(i)}$$

(this quantity is called the marginal likelihood). Since we have that $\mathbf{f}^{(i)} \mid \{1, \ldots, t\} \sim \mathcal{N}(0, \Sigma_{1:t,1:t})$ and $\mathbf{z}_{1:t}^{(i)} \mid \mathbf{f}^{(i)} \sim \mathcal{N}\left(\mathbf{f}^{(i)}, \sigma_{noise}^2 \mathbf{I}\right)$, we can use Appendix A.4.2 to compute the log marginal likelihood:

$$\log p(\mathbf{z}_{1:t}^{(i)} \mid \{1, \ldots, t\}, \boldsymbol{\theta}) = -\frac{1}{2}(\mathbf{z}_{1:t}^{(i)})^\top (\Sigma_{1:t,1:t} + \sigma_{noise}^2 \mathbf{I})^{-1} \mathbf{z}_{1:t}^{(i)} - \frac{1}{2} \log |\Sigma_{1:t,1:t} + \sigma_{noise}^2 \mathbf{I}| - \frac{t}{2} \log 2\pi.$$

Each term has an interpretation: the data fit term is $-\frac{1}{2}(\mathbf{z}_{1:t}^{(i)})^\top (\Sigma_{1:t,1:t} + \sigma_{noise}^2 \mathbf{I})^{-1} \mathbf{z}_{1:t}^{(i)}$, while $\frac{1}{2} \log |\Sigma_{1:t,1:t}|$ is a complexity penalty, and $\frac{t}{2} \log 2\pi$ is the normalization constant.

We set the hyperparameters by maximizing the log marginal likelihood using the partial derivatives with respect to the hyperparameters θ_j:

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{z}_{1:t}^{(i)} \mid \{1, \ldots, t\}, \boldsymbol{\theta}) = -\frac{1}{2}(\mathbf{z}_{1:t}^{(i)})^\top \tilde{\Sigma}_{1:t,1:t}^{-1} \frac{\partial(\tilde{\Sigma}_{1:t,1:t})}{\partial \theta_j} \tilde{\Sigma}_{1:t,1:t}^{-1} \mathbf{z}_{1:t}^{(i)} - \frac{1}{2} \mathrm{tr}\left(\tilde{\Sigma}_{1:t,1:t}^{-1} \frac{\partial(\tilde{\Sigma}_{1:t,1:t})}{\partial \theta_j}\right)$$

where $\tilde{\Sigma}_{1:t,1:t} = \Sigma_{1:t,1:t} + \sigma_{noise}^2 \mathbf{I}$. Using these derivatives, we can perform standard optimization routines in order to maximize the hyperparameters. Importantly, we can compare log marginal likelihood values for *different kernel functions* and for kernel functions with different hyperparameter values.

2.3.2.3 Gaussian Process Kernels as Pedestrian Dynamics Models

We describe our particular choice of kernel function in this section (up to the hyperparameters, which are trained using the methods outlined above). Because of the nature of our application (humans walking through a cafeteria), and the way that we modeled portions of agent trajectories (see Section 3.1.2), we had *a priori* insight about which kernel functions were appropriate. For instance, we were able to rule out the *squared exponential* covariance function

$$k_{SE}(t,t') = \exp\left(-\frac{(t-t')^2}{2\ell_{SE}^2}\right)$$

because the functions it encodes are strongly nonlinear. Instead, we chose to model pedestrian dynamics as the summation of a linear kernel (the nominal movement mode of humans between waypoints is linear)

$$k_{linear}(t,t') = t \cdot t' + \frac{1}{\gamma_{linear}^2},$$

a Matern kernel (it captures mild curving in the trajectory, common to pedestrian dynamics)

$$k_{Matern}(t,t') = s_{Matern} \cdot \left(1 + \frac{\sqrt{5}(t-t')}{\ell_{Matern}} + \frac{5(t-t')^2}{3\ell_{Matern}^2}\right) \exp\left(-\frac{\sqrt{5}(t-t')}{\ell_{Matern}}\right),$$

and a noise kernel (to account for sensor measurement noise)

$$k_{noise}(t,t') = \sigma_{noise}^2 \delta(t,t'),$$

where $\delta(t,t') = 1$ if $t = t'$ and is zero otherwise. Thus, our final kernel was

$$k^{(i)}(t,t') = s_{Matern} \cdot \left(1 + \frac{\sqrt{5}(t-t')}{\ell_{Matern}} + \frac{5(t-t')^2}{3\ell_{Matern}^2}\right) \exp\left(-\frac{\sqrt{5}(t-t')}{\ell_{Matern}}\right) + t \cdot t' + \frac{1}{\gamma_{linear}^2} + \sigma_{noise}^2 \delta(t,t').$$

(Figure 3.1 presents an actual human trajectory exhibiting each of these behavior modes: we observe linear and curvy motion, and noise in the measurements). Thus, four hyperparameters had to be learned: $s_{Matern}, \ell_{Matern}, \gamma_{linear}$ and σ_{noise}. We used the methods detailed in Section 2.3.2.2 to train these parameters from sample trajectories.

We point out that, in the absence of *a priori* information about what kernel function should be used, the methods of Section 2.3.2.2 can be used to compare different candidate kernel functions (e.g., squared exponential versus linear), since the values of the log marginal likelihoods can be compared *across* different kernel functions. Additionally, comparing marginal likelihood values can be used to guard against local minima when optimizing a fixed kernel function: for instance, one might randomly restart the hyperparameter optimization multiple times, and compare the marginal likelihood for the specific hyperparameter values found for each run, and then choose the most likely

hyperparameter set.

2.4 The Freezing Robot Problem

In Section 1.3, the *freezing robot problem* was presented as motivation for the methods developed in this thesis. In this section we provide mathematical details of the freezing robot problem. We also provide navigation methodologies for solution of the freezing robot problem that would be useful for environments like those shown in Figure 2.2.

Figure 2.2: Crowded cafeteria at the California Institute of Technology. Because of the many "goal locations" (the pizza bar, the soda fountain, the buffet, etc), distinct traffic currents only rarely materialize. Instead, movement patterns are highly turbulent, as people cross from left to right and top to bottom in a frenzied attempt to grab lunch before their next class. The density and complexity of the crowd ebbs as time moves away from 12pm, allowing for a vast diversity of experiments.

2.4.1 Mathematical Details of the Freezing Robot Problem

Consider agent i, where the index i can take values in the set $\{R, 1, 2, \ldots, n\}$, such that $\{1, 2, \ldots, n\}$ are human agents and $i = R$ is a robot. Suppose we have a distribution $p\left(\mathbf{f}^{(i)}\right)$ over each agent's trajectory

$$\mathbf{f}^{(i)} = \left(\mathbf{f}^{(i)}(1), \ldots, \mathbf{f}^{(i)}(T)\right)$$

over T timesteps, where each $\mathbf{f}^{(i)}(t) = (x_t, y_t) \in \mathbb{R}^2$ is the planar location of agent i at time t. We also have a likelihood function $p(\mathbf{z}_t^{(i)} \mid \mathbf{f}^{(i)}(t))$ for our observations. Importantly, since we are dealing with the case of multiple agents, we let

$$\mathbf{z}_{1:t} = (\mathbf{z}_{1:t}^{(1)}, \mathbf{z}_{1:t}^{(2)} \ldots, \mathbf{z}_{1:t}^{(n)}),$$

acknowledging that for some times t', we may not observe agent i, in which case $\mathbf{z}_{t'}^{(i)} = \varnothing$. We also remark that this notation is consistent with Section 2.3.1, since for linear-Gaussian random variables estimated by a Kalman filter, one can merely augment the state vector with the additional agents. Figure 2.3 illustrates these quantities.

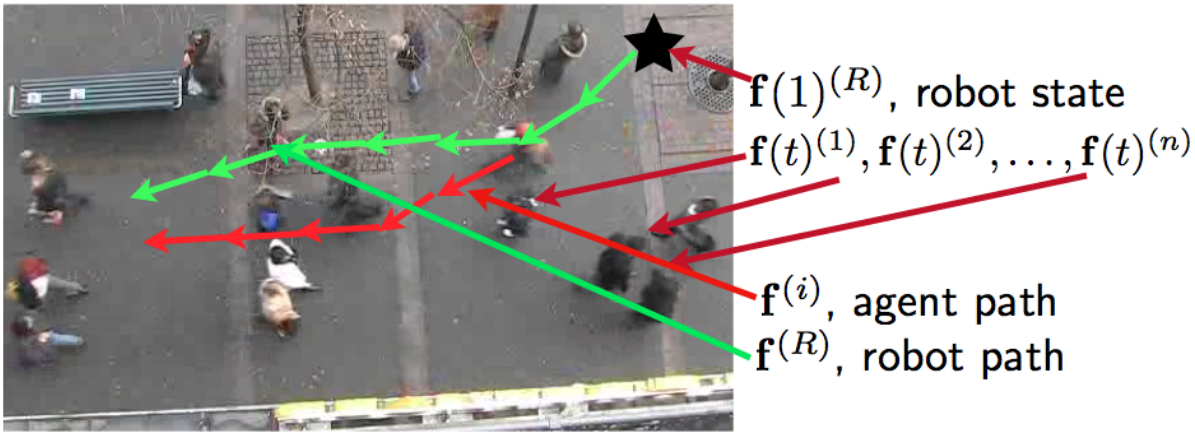


Figure 2.3: Illustration of the problem and the quantities we are interested in.

In the following, we will assume that data association is solved. Note that an observation of agent i is not necessarily independent of the robot's actions. For instance, if the robot's movement influences another agent's movement, then that observation *explicitly* depends on the robot's actions.

Our goal in dynamic navigation is to pick a policy $\boldsymbol{\pi}$ (see Section 2.1) that adaptively chooses a path $\mathbf{f}^{(R)}$ for the robot based on the observations $\mathbf{z}_{1:t}$ and any ancillary information (such as agent goal location, boundary locations, etc). The policy $\boldsymbol{\pi}$ is typically specified by stating the *next* location $\mathbf{f}^{(R)}(t+1)$ the robot should choose given all observational and ancillary information.

Thus, for any complete sequence of observations $\mathbf{z}_{1:T}$, the robot can potentially end up choosing a different path $\mathbf{f}^{(R)} = \boldsymbol{\pi}(\mathbf{z}_{1:T})$. The cost $J(\boldsymbol{\pi})$ of a policy $\boldsymbol{\pi}$ is the expected cost

$$J(\boldsymbol{\pi}) = \int p(\mathbf{f}, \mathbf{z}_{1:T}) c(\pi(\mathbf{z}_{1:T}), \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)}) d\mathbf{f} d\mathbf{z}_{1:T},$$

where, for a fixed robot trajectory $\mathbf{f}^{(R)}$, the cost function $c(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)})$ models the length of the path plus penalties for colliding with any of the agents. We use the shorthand notation $\mathbf{f} = (\mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)})$.

Unfortunately, solving for the optimal policy $\boldsymbol{\pi}$ requires solving a continuous-state *Markov deci-*

sion process (MDP), where the dimensionality grows linearly with the number of agents, which is intractable. Intuitively, the intractability is a consequence of attempting *exhaustive* enumeration; in the above expected cost $J(\boldsymbol{\pi})$, we are attempting to search over the policy space for all possible measurement sequences.

This insolubility is fairly common. In the path planning community, a state of the art, tractable approximation to this MDP is a method called *receding horizon control* (RHC), introduced and explained in Section 2.2. RHC proceeds in a manner similar to MDPs, albeit online: as observations become available, RHC calculates, based on some cost function, the optimal *non-adaptive* action (i.e., fixed path) to take at that time. Indeed, if we let $J(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t})$ be the objective function that calculates the "cost" of each path $\mathbf{f}^{(R)}$ based on the observations $\mathbf{z}_{1:t}$, that is

$$J(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) = \int c(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)}) p(\mathbf{f} \mid \mathbf{z}_{1:t}) d\mathbf{f},$$

where $\mathbf{f}^{(R)}$ is the trajectory of the robot, then RHC finds $\mathbf{f}_t^{(R*)}$, where

$$\mathbf{f}_t^{(R*)} = \arg\min_{\mathbf{f}^{(R)}} J(\mathbf{f}^{(R)} | \mathbf{z}_{1:t}).$$

As each new observation $\mathbf{z}_\tau$ arrives, for $\tau > t$, a new path $\mathbf{f}_\tau^{(R*)}$ is calculated and executed until another observation arrives.

Unfortunately, certain assumptions about the distribution $p(\mathbf{f} \mid \mathbf{z}_{1:t})$ can cause the minimum value of the objective function $J(\mathbf{f}^{(R*)} | \mathbf{z}_{1:t})$ to increase without bound as the number of agents n increases. This behavior of the objective function is what we call the *freezing robot problem* (see Figure 2.4(d) for an illustration).

To prove this, consider the special case of the objective function where we have *perfect knowledge* of how each pedestrian navigates through the space—that is, we have access to the true state trajectory $\bar{\mathbf{z}}_{1:T}$:

$$p(\bar{\mathbf{z}}_{1:T} \mid \mathbf{f}_{1:T}) = \delta(\bar{\mathbf{z}}_{1:T} - \mathbf{f}_{1:T}), \tag{2.1}$$

where $\mathbf{f}_{1:T}^{(i)} = [\mathbf{f}^{(i)}(1), \mathbf{f}^{(i)}(2), \ldots, \mathbf{f}^{(i)}(T)]$, and $\delta(a - b)$ is the Dirac delta function centered at a. For

this special case, we can compute the objective function:

$$\begin{aligned} J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:T}) &= \int c(\mathbf{f}_{1:T}^{(R)}, \mathbf{f}_{1:T}^{(1)}, \ldots, \mathbf{f}_{1:T}^{(n)}) p(\mathbf{f}_{1:T} \mid \bar{\mathbf{z}}_{1:T}) d\mathbf{f} \\ &\propto \int c(\mathbf{f}_{1:T}^{(R)}, \mathbf{f}_{1:T}^{(1)}, \ldots, \mathbf{f}_{1:T}^{(n)}) p(\bar{\mathbf{z}}_{1:T} \mid \mathbf{f}_{1:T}) p(\mathbf{f}_{1:T}) d\mathbf{f} \\ &= \int c(\mathbf{f}_{1:T}^{(R)}, \mathbf{f}_{1:T}^{(1)}, \ldots, \mathbf{f}_{1:T}^{(n)}) \delta(\bar{\mathbf{z}}_{1:T} - \mathbf{f}_{1:T}) p(\mathbf{f}_{1:T}) d\mathbf{f} \\ &= c(\mathbf{f}_{1:T}^{(R)}, \bar{\mathbf{z}}_{1:T}^{(1)}, \ldots, \bar{\mathbf{z}}_{1:T}^{(n)}). \end{aligned}$$

We thus see that as n increases, the minimum value of $J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:T})$ must also increase—the area remains fixed, while more of the free space becomes occupied (essentially, the planner must find a free path through existing open space, with the agent trajectories already traced out).

As the number of agent trajectories increases and thus fills out the fixed navigation area, the minimum value of $J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:T})$ increases without bound. If we reintroduce missing measurements and measurement uncertainty, the minimum value of $J(\mathbf{f}_{1:T}^{(R)}, n \mid \mathbf{z}_{1:T})$ can only be larger than $J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:T})$ for increasing values of n, since adding uncertainty places nonzero probability of agent occupation over a larger portion of the space. Thus, $J(\mathbf{f}_{1:T}^{(R)}, n \mid \mathbf{z}_{1:T})$ also increases without bound as n increases.

Additionally, if we consider the RHC case with perfect observations through time $t < T$, then the *predictive* objective function $J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:t})$ places nonzero probability of agent occupation over a larger portion of the space for $t' > t$ (similar to the case above). Furthermore, if we relax the objective function to the standard RHC case with measurement uncertainty $J(\mathbf{f}_{1:T}^{(R)}, n \mid \mathbf{z}_{1:t})$, we further spread uncertainty over times $t' \leq t$ that have already been observed. Therefore, we have that both $J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:t})$ and $J(\mathbf{f}_{1:T}^{(R)}, n \mid \mathbf{z}_{1:t})$ can only be larger than $J(\mathbf{f}_{1:T}^{(R)}, n \mid \bar{\mathbf{z}}_{1:T})$, and so RHC also exhibits freezing robot behavior.

2.4.2 Approaches for Solving the Freezing Robot Problem

In order to fix the freezing robot problem, nearly all state of the art approaches (see Section 1.2) focus on *individual* agent prediction. In particular, Du Toit [28] anticipates the observations (effectively assuming that a certain measurement sequence of the *entire* trajectory sequence has already taken place at time $t < T$); the approach is motivated by the assumption that the culprit of the freezing robot problem is an uncertainty explosion, illustrated in Figure 2.4(c) and discussed in Section 1.3. The claim is that if you can control the covariance, then you can keep the minimum value of $J(\mathbf{f}^{(R)}|\mathbf{z}_{1:t})$ low for moderately dense crowds, and thus solve the freezing robot problem (other approaches, which incorporate more accurate agent modeling, are similar in motivation, since better dynamic models would reduce predictive covariance as well). However, as shown above, approaches that work at improving the independent agent prediction or reducing the covariance only solve the

freezing robot problem for crowd densities below a certain threshold; importantly, they cannot be expected to solve the freezing robot problem *in general*, no matter how favorable the circumstances (even for the case of perfect knowledge of the future).

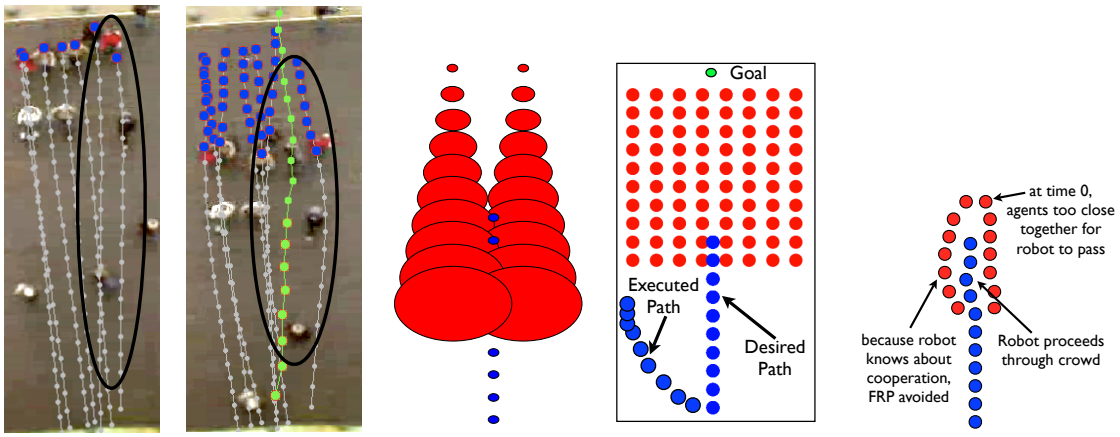


Figure 2.4: **(a,b)** Empirical evidence of joint collision avoidance: blue circles (representing current position) over gray lines are pedestrians moving down, black circles are the area of interest, and green dots are a pedestrian moving upwards. In (a), the blue pedestrians have not yet seen the green person; their projected trajectories (in gray) continue shoulder to shoulder. In (b), green dots surrounded by red circles are the current position of the pedestrian moving up, and *all* of the pedestrians have adjusted their trajectories to create space—notice how wide the gray prediction has become. It is this joint collision avoidance behavior that we advocate in this thesis. **(c-e)** Illustration of freezing robot problem. Dynamic crowd agents in red traveling downward, robot we are trying to control in blue. The multiple dots indicate multiple points along one trajectory. (c) Uncertainty explosion due to uncorrected prediction. (d) Even with perfect prediction, room for robot navigation may not exist. (e) Modeling cooperative collision avoidance remedies the freezing robot problem.

This analysis suggests that the planning problem, as described above, is ill-posed. We thus revisit our probability density,

$$p(\mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}),$$

and remark that a crucial element is missing—the agent motion model is agnostic of the navigating robot. One solution is thus immediately apparent: include an interaction between the robots and the agents (in particular, a joint collision avoidance) in order to lower the cost. We additionally remark that in the illustration in Figure 2.4(b), the crowd experiments catalogued in the research of Helbing and Molnar [46], Helbing et al. [48, 47], the multi-robot coordination theorems of van den Berg et al. [115, 116], and the tracking experiments of Pellegrini et al. [79, 80] and Luber et al. [70], all corroborate the argument that autonomous dynamic agents utilize joint collision avoidance behaviors for successful crowd navigation. We thus consider methods to incorporate such an interaction.

We motivate this discussion by first extending the likelihood of Equation 2.1 to include the robot

trajectory $\mathbf{f}_{1:T}^{(R)}$:

$$p(\bar{\mathbf{z}}_{1:T}, \bar{\mathbf{z}}_{1:T}^{(R)} \mid \mathbf{f}_{1:T}, \mathbf{f}_{1:T}^{(R)}) = \delta([\bar{\mathbf{z}}_{1:T}, \bar{\mathbf{z}}_{1:T}^{(R)}] - [\mathbf{f}_{1:T}, \mathbf{f}_{1:T}^{(R)}]).$$

Notice that because we have coupled the robot's trajectory $\mathbf{f}^{(R)}$ with the agent trajectories $\mathbf{f}$ we can now (literally) vary the future joint configurations $[\bar{\mathbf{z}}_{1:T}, \bar{\mathbf{z}}_{1:T}^{(R)}]$, and, in turn, vary the minimum value of the objective function (whereas for Equation 2.1 we were locked into a minimum value that was based on a *fixed* value of $\bar{\mathbf{z}}_{1:T}$). As explained above, approaches that do not consider cooperation between the robot and the agents fail as soon as the navigation area fills with trajectories (if the robot is not influencing $\bar{\mathbf{z}}_{1:T}$, it is unlikely that a free path will emerge randomly); by incorporating cooperation, however, we are able to manipulate the responses of the agents (i.e., their trajectories) and thus to *create space*.

We discuss two ways that human-robot interaction (or human-robot cooperation) may be modeled. One approach to modeling this interaction would be to use a *conditional* density $p(\mathbf{f} \mid \mathbf{z}_{1:t}, \mathbf{f}^{(R)})$, that encodes assumptions on how the agents react to the robot's actions, i.e., the idea that all agents will "give way" to the robot's trajectory. The problem with this approach is that it assumes that the robot has the ability to fully control the crowd. Thus, this approach would not only create an obnoxious robot, but an overaggressive and potentially dangerous one as well. This method is unsuitable for crowded situations.

The other alternative, which we advocate in this thesis, is to model the robot as one of the agents, and subsequently model a joint distribution describing their interaction:

$$p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}).$$

This distribution encodes the idea of cooperative planning (e.g., *cooperative* collision avoidance) by treating robot and agent behaviors as equivalent (unlike the conditional density, where the robot was given priority, or the noncooperative density $p(\mathbf{f} \mid \mathbf{z}_{1:t})$, where the agents were given priority). We point out an important characteristic of this formulation. Although the robot *anticipates* agent cooperation, the data ultimately takes precedence. Consider the situation where an agent does not cooperate with the robot (perhaps the agent does not see the robot, or perhaps the agent just does not want to cooperate). As the robot approaches this agent, it will predict that the agent will eventually act to cooperatively create space. However, as the robot moves closer to the agent, the evidence (and thus the prediction) that the agent is not going to cooperate will outweigh the prior belief that agent will cooperate. Thus, the robot will compensate, maneuvering around the unyielding agent (we observe this behavior in Chapter 5).

With the joint model $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$, planning corresponds to computing $\arg\max_{(\mathbf{f}^{(R)}, \mathbf{f})} p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$, i.e., inferring what the robot *should* do given observations of the other agents (this approach is an example of "reducing planning to inference", that we discuss in Section 3.3).

Finally, we remark that the formulation $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ is independent of the completeness of the data—that is, $\mathbf{z}_{1:t}$ can range from being globally complete (i.e., we are given deterministic access to each agent's state at each time t'), to local observations (e.g., an "onboard" sensor that only observes agents within the line of sight of the robot), to complete data outage. As data reliability decreases, navigation performance will accordingly degrade, but the method does not require perfect information.

Chapter 3

Interacting Gaussian Processes

We begin this chapter by deriving *interacting Gaussian processes* for crowd prediction. We then describe how approximate inference is performed on the interacting Gaussian processes density, followed by a discussion of "the navigation density", or how robotic navigation in dense human crowds can be interpreted as a statistic of the interacting Gaussian processes density. We conclude with the results of a simulation experiment.

3.1 Crowd Prediction Modeling with Interacting Gaussian Processes

In this section we introduce *interacting Gaussian processes* (IGP). Although we ultimately interpret this density for robot navigation, IGP is also a crowd prediction model. We begin by deriving individual models of goal driven human motion using mixtures of Gaussian processes. We then couple these individual models with the *interaction potential*.

3.1.1 Gaussian Processes for Modeling Single Goal Trajectories

An advantage to the Gaussian Process formalism (introduced in Section 2.3.2) is that it estimates entire trajectories. This allows us to incorporate a single goal $\mathbf{g}$ (known up to Gaussian uncertainty, $\mathbf{g} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{g}}, \sigma_{\mathbf{g}}^2 \mathbf{I})$) such that the resulting distribution over trajectories reflects the full impact of the additional data (Figure 2.1(b) illustrates this idea). Implementation wise, we merely treat the goal information as a measurement on the final step of the trajectory. That is, having observed agent i for t steps, we can augment $\mathbf{z}_{1:t}^{(i)}$ with $\mathbf{z}_T^{(i)} = \mathbf{g}$. We now update our Gaussian Process using $\mathbf{z}_{1:t,T}^{(i)} = [\mathbf{z}_{1:t}^{(i)} \ \mathbf{z}_T^{(i)}]$, arriving at $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t,T}^{(i)}) = GP(m_{t,T}^{(i)}, k_{t,T}^{(i)})$, where

$$
\begin{aligned}
m_{t,T}^{(i)}(t') &= \Sigma_{1:t,t'}^{\mathrm{T}} (\Sigma_{1:t,1:t} + \tilde{\mathbf{I}})^{-1} \mathbf{z}_{1:t,T}^{(i)} \\
k_{t,T}^{(i)}(t_1, t_2) &= k(t_1, t_2) - \Sigma_{1:t,t_1}^{\mathrm{T}} (\Sigma_{1:t,1:t} + \tilde{\mathbf{I}})^{-1} \Sigma_{1:t,t_2},
\end{aligned}
$$

and

$$\tilde{\mathbf{I}} = \text{diag}[\sigma^2_{noise}, \sigma^2_{noise}, \ldots, \sigma^2_{noise}, \sigma^2_{\mathbf{g}}].$$

By varying the amount of noise associated with this goal measurement, we can encode how certain we are about the goal. Waypoints along trajectories can be easily encoded in the same manner; we exploit this flexibility in Section 3.1.2.2 to design appropriate models for multi-destination behavior.

For the special case of the *robot's goal*, $\mathbf{z}_T^{(R)}$, we set the noise differently. The value of $\sigma^2_{\mathbf{z}_T^{(R)}}$ reflects how precise the robot must be in executing its task. For example, if a robotic arm is to insert a rivet in an aircraft wing, then $\sigma^2_{\mathbf{z}_T^{(R)}}$ will be quite small, to reflect aircraft tolerances. If a robot needs to travel across a busy cafeteria to deliver plates to a human being, then $\sigma^2_{\mathbf{z}_T^{(R)}}$ can be quite large, perhaps on the order of 1/2 meter or so.

Alternatively, we can *derive* how goal information should be included using marginalization followed by the chain rule of probability (see Appendix A.1); this more cumbersome approach will prove important when we generalize to the case of multiple goals in Section 3.1.2. In particular, we sum over all possible discrete goal indices $m \in \mathbb{N}^+$ for the goal variable $\bar{\mathbf{g}}_m$; for each m, we integrate over all possible goal arrival times $\bar{T}_m \in \mathbb{R}^+$:

$$\begin{aligned}
p(\mathbf{f}^{(i)} \mid \mathbf{z}^{(i)}_{1:t}) &= \sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)}, (\bar{\mathbf{g}}_m, \bar{T}_m) \mid \mathbf{z}^{(i)}_{1:t}) \\
&= \sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)} \mid \mathbf{z}^{(i)}_{1:t}, (\bar{\mathbf{g}}_m, \bar{T}_m)) p((\bar{\mathbf{g}}_m, \bar{T}_m) \mid \mathbf{z}^{(i)}_{1:t}) \\
&= \sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)} \mid \mathbf{z}^{(i)}_{1:t}, (\bar{\mathbf{g}}_m, \bar{T}_m)) \delta\left((\bar{\mathbf{g}}_m, \bar{T}_m) - (\mathbf{g}, T)\right) \\
&= p(\mathbf{f}^{(i)} \mid \mathbf{z}^{(i)}_{1:t}, (\mathbf{g}, T)).
\end{aligned}$$

where $\delta\left((\bar{\mathbf{g}}_m, \bar{T}_m) - (\mathbf{g}, T)\right)$ is 1 when $\bar{\mathbf{g}}_m = \mathbf{g}$ and $\bar{T}_m = T$ but zero otherwise. Furthermore, $p((\mathbf{g}, T) \mid \mathbf{z}_{1:t}) = \delta\left((\bar{\mathbf{g}}_m, \bar{T}_m) - (\mathbf{g}, T)\right)$ since we assume knowledge of *which* goal the agent is going to ($\mathbf{g}$) and how long it will take to arrive (T). In the final step, we integrate over the variables in the delta function and recover a distribution conditioned on $(\mathbf{g}, T)$. Bear in mind that we integrated over the possible goals (see Section 3.1.2.1), *not* the uncertainty surrounding the *location* of a certain goal. Indeed, we still assume that goal location is only known up to Gaussian uncertainty, $\mathbf{g} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{g}}, \sigma^2_{\mathbf{g}}\mathbf{I})$, and so, if we still wish to model the trajectory prior using Gaussian processes—that is, $\mathbf{f}^{(i)} \sim GP(0, k^{(i)})$—then we recover

$$p(\mathbf{f}^{(i)} \mid \mathbf{z}^{(i)}_{1:t}) = GP(m^{(i)}_{t,T}, k^{(i)}_{t,T}).$$

We will use this marginalization-chain rule approach to model situations where we only have incomplete information about the goal and the time that it will take an agent to reach the goal.

3.1.2 Gaussian Process Mixtures for Modeling Multiple Goal Trajectories

In practice there may be uncertainty between multiple, discrete goals that an agent could pursue (Figure 3.1); similarly, it is exceedingly rare to know in advance the time it takes to travel between these waypoints. For these reasons, we introduce a novel probabilistic model over waypoints and the transition time between these waypoints. The motion model is then a mixture of Gaussian processes interpolating between these waypoints.

Figure 3.1: An example trajectory of a cafeteria patron. The trajectory was hand labeled and segmented; blue dots are part of the nominal trajectory (modeled with the kernel function $k = k_{linear} + k_{matern} + k_{noise}$, as in Section 2.3.2.3), green dots are goals (see Section 3.1.2.2), and red represents interaction between agents (see Section 3.1.3)

.

3.1.2.1 Definitions

We begin with the assumption that the environment in which we will be doing trajectory prediction has a *fixed* number of goals G (corresponding roughly to the number of eating stations in the cafeteria):

$$\mathbf{g} = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_G)$$

For the purposes of this analysis, we restrict the distributions governing each goal random variable to be Gaussian. We also restrict our goals $\mathbf{g}_k$ to lie in the plane $\mathbb{R}^2$.

Using data from the cafeteria—the cafeteria floor was first divided into a grid, and then the amount of time a bin was occupied by a person was collected—we used Gaussian mixture model clustering (Bishop [14]) to segment the pedestrian track data into "hot spots". In particular, we learned

$$p(\mathbf{g}) = \sum_{k=1}^{G} \beta_k \mathcal{N}\big(\mathbf{g}_k; \boldsymbol{\mu}_{\mathbf{g}_k}, \boldsymbol{\Sigma}_{\mathbf{g}_k}\big).$$

where β_k is the weight of each component learned, $\boldsymbol{\mu}_{\mathbf{g}_k}$ is the mean of the goal location, and $\boldsymbol{\Sigma}_{\mathbf{g}_k}$ is the uncertainty around the goal. Figure 3.2 plots $p(\mathbf{g})$ on top of an image of the cafeteria. Notice that the hotspots occur around the perimeter of the cafeteria, where the food is served.

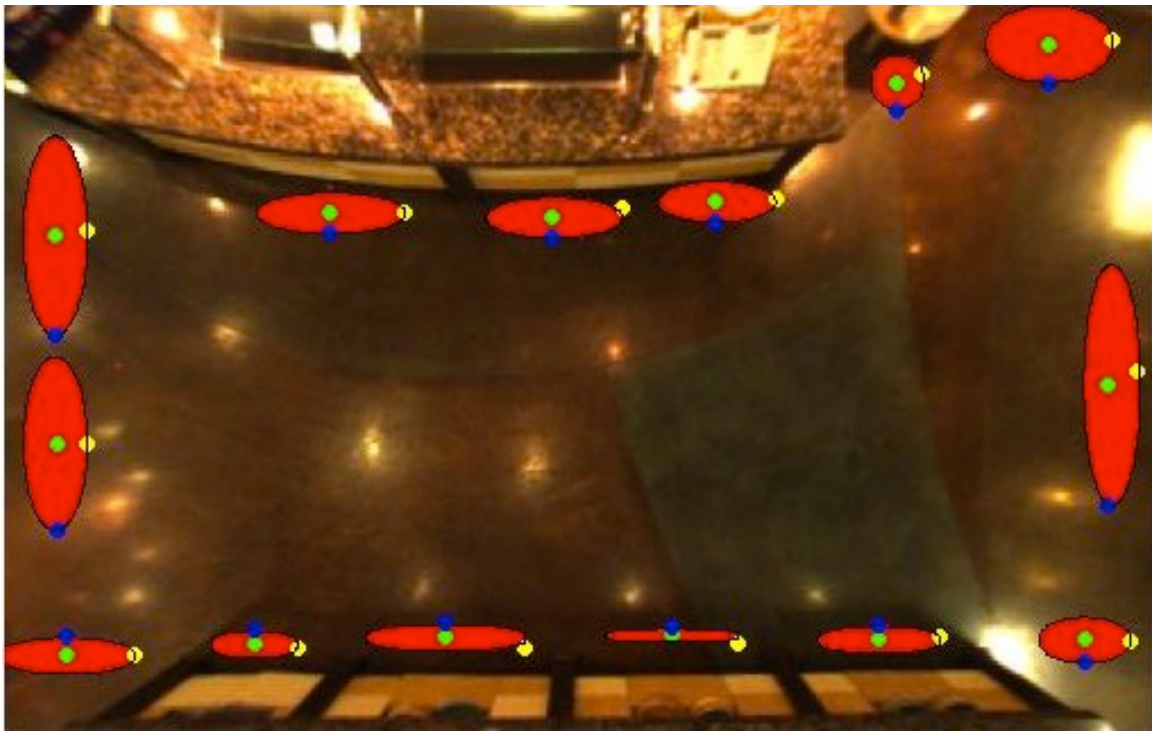


Figure 3.2: The Gaussian goals identified in the cafeteria. The green dots represent the mean, and the blue and yellow circles represent the length of the covariance axes.

Once we have clustered the data into the distribution $p(\mathbf{g})$, we then need to understand how agents move *between* these goals. For instance, a patron might first go to the pizza station, followed by the soda fountain, followed by checking out at the cashier bench. We therefore define the transition probability $p(\mathbf{g}_a \rightarrow \mathbf{g}_b)$ for all $a, b \in \{1, 2, \ldots, G\}$. These transition probabilities are empirically determined from experimental data. For every transition between two goals $\mathbf{g}_a \rightarrow \mathbf{g}_b$ we define the duration random variable $T_{a\rightarrow b}$, which is governed by a density $p(T_{a\rightarrow b})$ that we also determine empirically.

Finally, we introduce a waypoint sequence

$$\bar{\mathbf{g}}_m = (\mathbf{g}_{m_1} \rightarrow \mathbf{g}_{m_2} \rightarrow \cdots \rightarrow \mathbf{g}_{m_F})$$

(where $\mathbf{g}_{m_k}$ is a waypoint, and $m_k \in \{1, 2 \ldots, G\}$) for locations indexed by $m_1, m_2, \ldots, m_F$ where $F \in \mathbb{N}^+$, with associated way point durations

$$\bar{T}_m = \{T_{m_0 \to m_1}, T_{m_1 \to m_2}, \cdots, T_{m_{F-1} \to m_F}\}$$

where $T_{m_0 \to m_1}$ is the time to the first goal.

3.1.2.2 Generative Process for a Sequence of Waypoints

We now describe a generative process for a sequence of waypoints $\bar{\mathbf{g}}_m$. Beginning with the set of learned goals $\mathbf{g}$, we draw indices from the set $\{1, 2, \ldots G\}$. The first index is drawn uniformly at random[1], with the following indices drawn according to the transition probability $p(\mathbf{g}_a \to \mathbf{g}_b)$. Simultaneously, we draw the transition times $T_{a \to b}$ according to the distribution $p(T_{a \to b})$. We stop sampling goal points of the sequence $\bar{\mathbf{g}}_m = (\mathbf{g}_{m_1} \to \mathbf{g}_{m_2} \to \cdots \to \mathbf{g}_{m_F})$ when $\sum_{j=0}^{F-1} T_{m_j \to m_{j+1}}$ exceeds the prediction horizon T. Notice that the value of F for a particular sequence will not necessarily match that of another sequence, since the time between goals $T_{m_j \to m_{j+1}}$ varies for all j.

Thus, we can formulate our prediction model for agent i such that we sum over all possible waypoint sequences $\bar{\mathbf{g}}_m$, and for each particular waypoint sequence $\bar{\mathbf{g}}_m$ we integrate out all possible associated durations $\bar{T}_m$:

$$p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) = \sum_{\bar{\mathbf{g}}_m} \left(\int_{\bar{T}_m} p(\mathbf{f}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}) \right).$$

Using the chain rule, we end up with

$$\begin{aligned} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) &= \sum_{\bar{\mathbf{g}}_m} \left(\int_{T_{m_0 \to m_1}} \int_{T_{m_1 \to m_2}} \cdots \int_{T_{m_{F-1} \to m_F}} p(\mathbf{f}^{(i)}, \bar{\mathbf{g}}_m, T_{m_0 \to m_1}, T_{m_1 \to m_2}, \cdots, T_{m_{F-1} \to m_F} \mid \mathbf{z}_{1:t}^{(i)}) \right) \\ &= \sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m) p(\bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}). \end{aligned} \tag{3.1}$$

Notice that for each goal sequence $\bar{\mathbf{g}}_m$, we potentially have a different number of waypoints $\mathbf{g}_{m_k}$.

3.1.3 Interacting Gaussian Processes

Our key modeling idea is to capture the dynamic interactions by introducing dependencies between the Gaussian processes. We begin with the independent Gaussian process models

$$p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}^{(R)}),\ p(\mathbf{f}^{(1)} \mid \mathbf{z}_{1:t}^{(1)}),\ \ldots,\ p(\mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}^{(n)}),$$

[1] We acknowledge that the parameters β_k from the mixture $p(\mathbf{g})$ could inform this initial sample. We chose instead to allow this initial waypoint to be weighted by measurements, as discussed in Section 3.2.2.

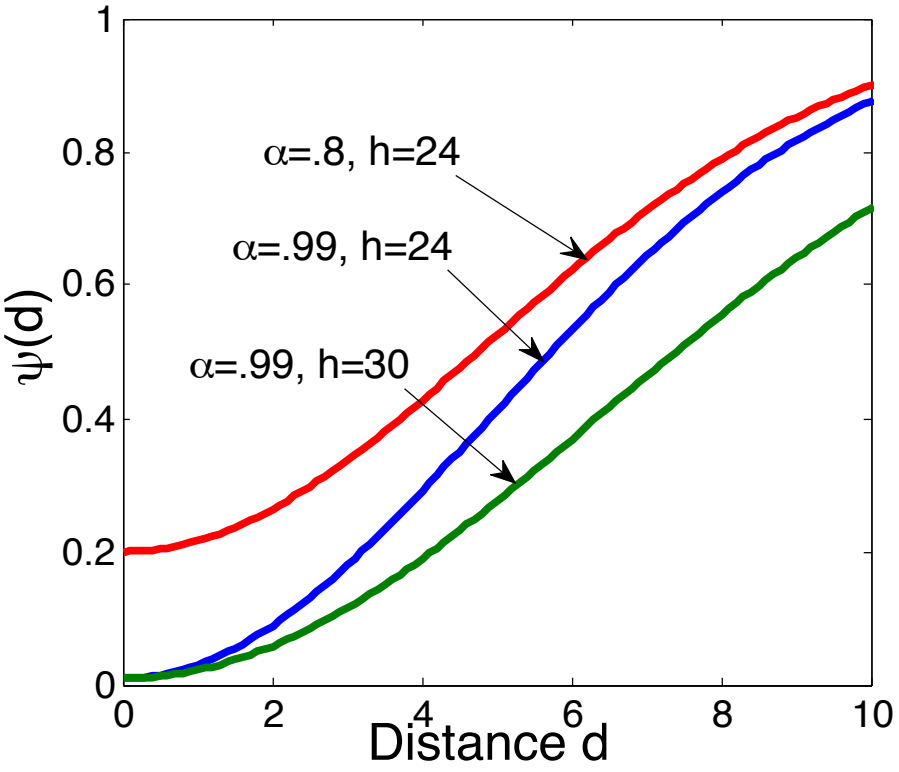


Figure 3.3: The interaction potential $1 - \alpha \exp(-\frac{1}{2h^2}d^2)$, for various α, h.

and couple them by multiplying in an *interaction potential*

$$\psi(\mathbf{f}^{(R)}, \mathbf{f}) = \psi(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)}),$$

where $\mathbf{f} = (\mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)})$. Thus,

$$p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t}) = \frac{1}{Z}\psi(\mathbf{f}^{(R)}, \mathbf{f}) \prod_{i=R}^{n} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}). \tag{3.2}$$

The product $\prod_{i=R}^{n}$ is meant to indicate that the robot is included in the calculation. In our experiments, we chose the interaction potential as:

$$\psi(\mathbf{f}^{(R)}, \mathbf{f}) = \prod_{i=R}^{n} \prod_{j=i+1}^{n} \prod_{\tau=t}^{T} \Big(1 - \alpha \exp\big(-\frac{1}{2h^2}|\mathbf{f}^{(i)}(\tau) - \mathbf{f}^{(j)}(\tau)|\big)\Big)$$

where $|\mathbf{f}^{(i)}(\tau) - \mathbf{f}^{(j)}(\tau)|$ is the Euclidean distance at time τ between agent i and agent j. The rationale behind our choice is that any specific instantiation of paths

$$\mathbf{f}_l^{(R)}, \mathbf{f}_l^{(1)}, \mathbf{f}_l^{(2)}, \ldots, \mathbf{f}_l^{(n)}$$

becomes very unlikely if, at any time τ, any two agents i and j are too close. Furthermore, the parameter h controls the "safety margin" of the repulsion, and $\alpha \in [0, 1]$ the strength of the repulsion. The parameter h was chosen to be the closest approach of two navigating pedestrians (in both simulation (Section 3.4) and in the dining hall experiments (Chapter 5)), while α was chosen to be in the range [0.9 0.99]. See Figure 3.3 for an illustration.

3.1.4 Multi-Goal Interacting Gaussian Processes

Importantly, we point out that if we expand the IGP density to take goal and waypoint duration uncertainty into account by using the motion mixture model approximation, then we have *multi-goal interacting Gaussian processes* (mgIGP):

$$\begin{aligned} p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}) &= \frac{1}{Z}\psi(\mathbf{f}) \prod_{i=1}^{n} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) \\ &= \frac{1}{Z}\psi(\mathbf{f}) \prod_{i=1}^{n} \left(\sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}) \right) \\ &\approx \frac{1}{Z}\psi(\mathbf{f}) \prod_{i=1}^{n} \left(\sum_{k=1}^{N_p} w_k^{(i)} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_k, \bar{T}_k) \right). \end{aligned}$$

3.2 Approximate Inference for Interacting Gaussian Processes

For Gaussian processes, exact and efficient inference is possible. However, the introduction of the interaction potential makes the posterior $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ non-Gaussian and thus approximate inference is required. Standard approaches to approximate inference in models derived from Gaussian processes include the Laplace approximation (Bishop [14]) and expectation propagation (Minka [75]). These methods approximate the non-Gaussian posterior by a Gaussian which has the same mode, or which minimizes the Kullback-Leibler divergence, respectively. These methods are most effective if the posterior is unimodal (and can be well-approximated by a Gaussian). With interacting Gaussian processes, however, the posterior is expected to be multimodal. In particular, for two agents moving towards each other in a straight line, evasion in either direction is equally likely. This is akin to people walking towards each other, flipping from one "mode" to the other while attempting to not collide.

To cope with the multimodality, we use an approximate inference technique based on importance sampling, a well understood approximate inference method for Bayesian statistics (for an introduction see Arulampalam et al. [7], Doucet et al. [26], Ristic et al. [88], Thrun et al. [111] or Herman [51]; for a more detailed, up to date analysis of the method see Doucet and Johansen [25], Andrieu et al. [2] or Koller and Friedman [59]).

In this section, we first describe importance sampling for the special case of interacting Gaussian

processes that have a single known goal for each agent (we call this "single mode Gaussian processes" in Section 3.2.1). We then generalize the importance sampling procedure for individual agent models that follow Equation 3.1, with multiple goals and unknown times to goal. That is, we employ two *different* sampling steps: first we compute (online) a sample based approximation of each agent's mixture process (Section 3.2.2)

$$p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) = \sum_{\bar{\mathbf{g}}_m} \Big(\int_{\bar{T}_m} p(\mathbf{f}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}) \Big),$$

and then we compute a sample based approximation of the full multi-goal interacting Gaussian processes posterior $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ in Section 3.2.3 .

3.2.1 Sample Based Approximation of Interacting Gaussian Processes

We implement importance sampling (see chapter 4 of Herman [51] for a discussion of importance sampling) for approximate inference of the single known goal interacting Gaussian process density as follows:

- For all agents i, sample independent trajectories of agent i from the prior (see Appendix A.4.4):

  $$(\mathbf{f}^{(i)})_l \sim p(\mathbf{f}^{(i)} | \mathbf{z}_{1:t}^{(i)}),$$

  where $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)})$ is the Gaussian process sampling density for agent trajectory i. A *joint sample* is the collection of $n+1$ such agent samples: $(\mathbf{f}^{(R)}, \mathbf{f})_l$ (see the left side of Figure 3.4 for an illustration of this idea). Since we are approximating the density $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$, the joint sample is our quantity of interest.

- Evaluate the weight of each sample $(\mathbf{f}^{(R)}, \mathbf{f})_l$ using the rules of importance sampling:

  $$\begin{aligned} \eta_l &= \frac{p((\mathbf{f}^{(R)}, \mathbf{f})_l \mid \mathbf{z}_{1:t})}{\prod_{i=R}^{n} p((\mathbf{f}^{(i)})_l \mid \mathbf{z}_{1:t}^{(i)})} \\ &= \frac{\psi((\mathbf{f}^{(R)}, \mathbf{f})_l) \prod_{j=R}^{n} p((\mathbf{f}^{(j)})_l \mid \mathbf{z}_{1:t}^{(j)})}{\prod_{i=R}^{n} p((\mathbf{f}^{(i)})_l \mid \mathbf{z}_{1:t}^{(i)})} \\ &= \psi((\mathbf{f}^{(R)}, \mathbf{f})_l). \end{aligned}$$

- The posterior is then approximated by the empirical sampling distribution,

  $$p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t}) \approx \frac{1}{\sum_{s=1}^{N} \eta_s} \sum_{l=1}^{N} \eta_l \delta([\mathbf{f}^{(R)}, \mathbf{f}]_l - [\mathbf{f}^{(R)}, \mathbf{f}]),$$

  where $\delta([\mathbf{f}^{(R)}, \mathbf{f}]_l - [\mathbf{f}^{(R)}, \mathbf{f}])$ is the delta function centered at sample $[\mathbf{f}^{(R)}, \mathbf{f}]_l$.

As we let the number N of samples grow, we approximate $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ to arbitrary accuracy. Note that all samples are independent of one another. Thus, the technique can be parallelized. See Figure 3.4 for an illustration of this process.

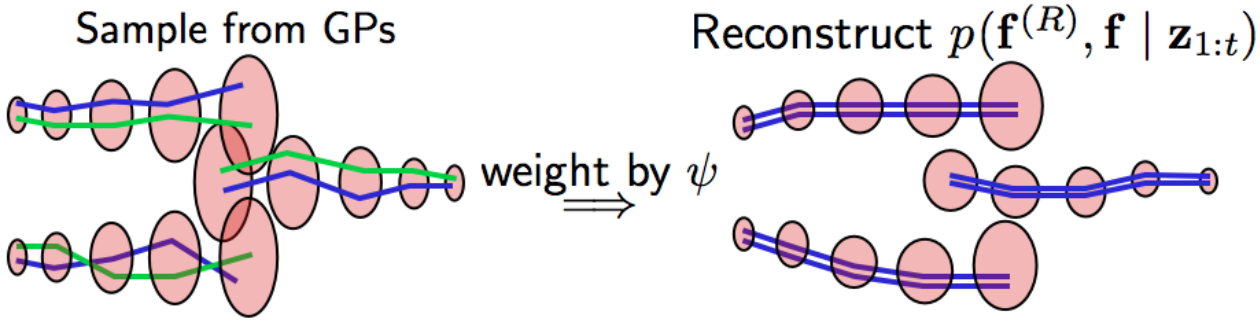


Figure 3.4: First samples $\mathbf{f}_l^{(i)}$ are drawn for each agent i. In this illustration, 3 agents are under consideration; we represent one joint sample of the 3 agents with 3 green lines, and another joint sample with 3 blue lines. The samples are then weighted and combined to produce an estimate of the interacting Gaussian processes density.

3.2.2 Sample Based Approximation of Gaussian Process Mixtures

Unfortunately, the expansion in Equation 3.1 is intractable, so we employ a sample based approximation for the distribution over goal waypoints and durations for agent i

$$p(\bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}) \approx \sum_{k=1}^{N_p} w_k^{(i)} \delta([\bar{\mathbf{g}}_m, \bar{T}_m]_k - [\bar{\mathbf{g}}_m, \bar{T}_m]),$$

where we utilize the empirically derived density

$$\left(\bar{\mathbf{g}}_m, \bar{T}_m\right)_k \sim p(\bar{\mathbf{g}}_m, \bar{T}_m).$$

Substituting $\sum_{k=1}^{N_p} w_k^{(i)} \delta\Big[(\bar{\mathbf{g}}_m, \bar{T}_m)_k - (\bar{\mathbf{g}}_m, \bar{T}_m)\Big]$ into Equation 3.1, we generate the approximation

$$\begin{aligned}
p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) &= \sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m) p(\bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}) \\
&= \sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m) \sum_{k=1}^{N_p} w_k^{(i)} \delta([\bar{\mathbf{g}}_m, \bar{T}_m]_k - [\bar{\mathbf{g}}_m, \bar{T}_m]) \\
&\approx \sum_{k=1}^{N_p} w_k^{(i)} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_k, \bar{T}_k).
\end{aligned}$$

The samples collapse the infinite sum of integrals to one finite sum. This finite component mixture process is illustrated in Figure 3.5.

In order to generate particles $(\bar{\mathbf{g}}_k, \bar{T}_k)$, we first draw a sequence of waypoints $\bar{\mathbf{g}}_k$ and then the

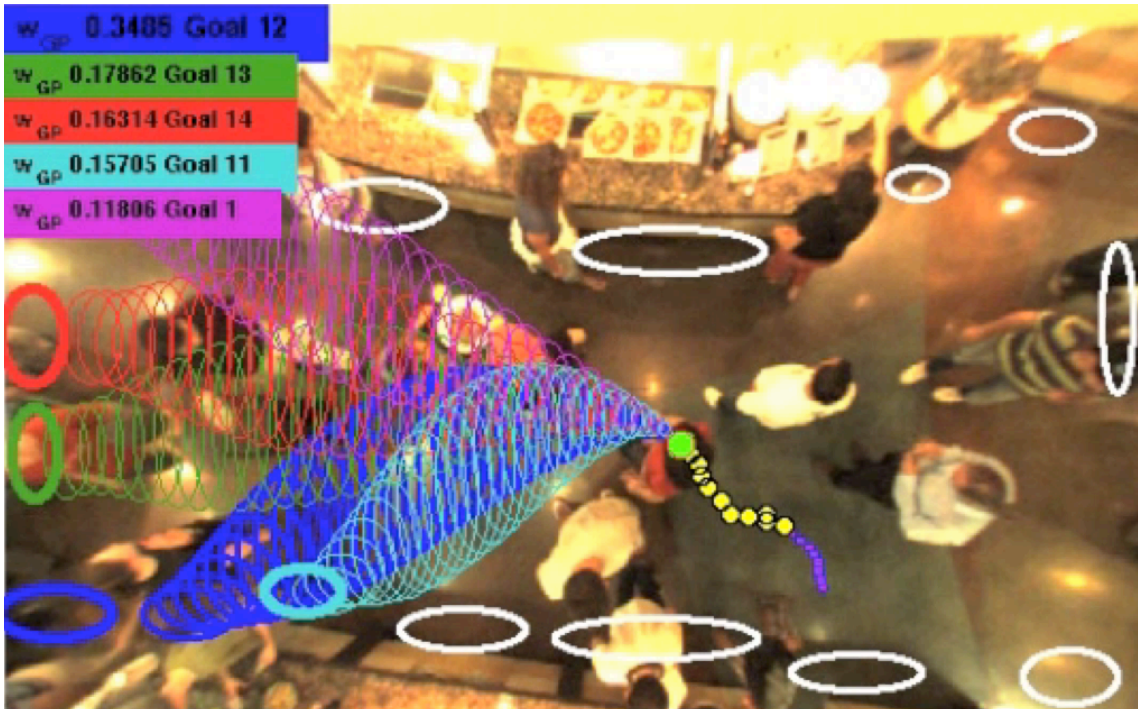


Figure 3.5: A patron moves through the cafeteria (solid green circle). Trailing yellow dots are history, and tubes are Gaussian Process mixture components. Gaussian Process mixture weights are in the upper left corner.

corresponding sequence of waypoint durations $T_{k_a \to k_b}$. To draw the waypoints, we begin by first sampling $\mathbf{g}_{k_1}$ uniformly from the G goals. We then draw $T_{k_0 \to k_1}$ according to a distribution with mean given by the average time to travel from the current point to $\mathbf{g}_{k_1}$. Then, $\mathbf{g}_{k_2}$ is drawn according to the transition probabilities $p(\mathbf{g}_{k_1} \to \mathbf{g}_b)$, and $T_{k_1 \to k_2}$ is consequently sampled. We continue until the sum of the duration waypoints reaches or exceeds T_{max}, and then drop the most recently sampled goal.

Additionally, we evaluate the individual mixture component weights according to

$$w_k^{(i)} = \frac{p(\left(\bar{\mathbf{g}}_m, \bar{T}_m\right)_k \mid \mathbf{z}_{1:t}^{(i)})}{p(\bar{\mathbf{g}}_m, \bar{T}_m)} \propto p(\mathbf{z}_{1:t}^{(i)} \mid \left(\bar{\mathbf{g}}_m, \bar{T}_m\right)_k)$$

that is, we evaluate the likelihood of the observed data $\mathbf{z}_{1:t}^{(i)}$ assuming a specific waypoint-duration pair $(\bar{\mathbf{g}}_m, \bar{T}_m)_k$. We illustrate this idea in Figure 3.6.

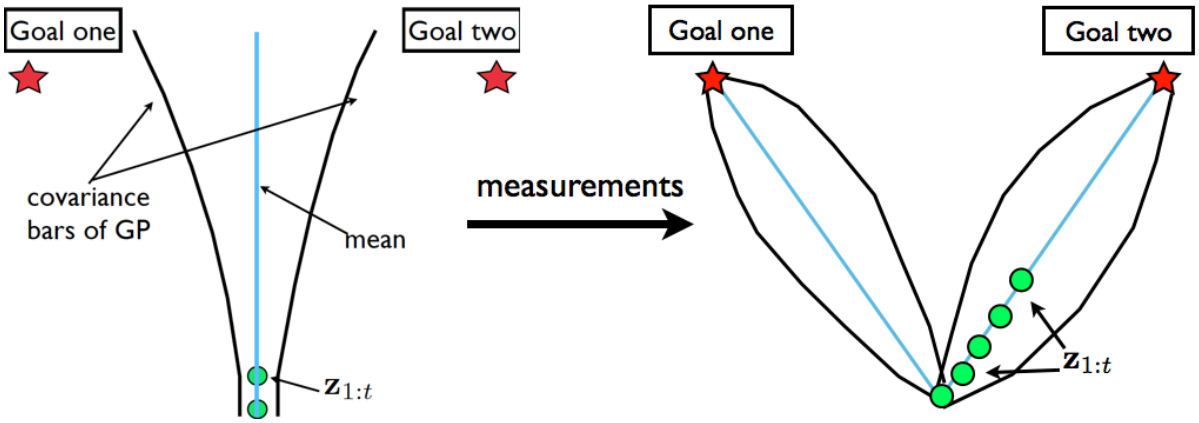


Figure 3.6: Certain goals (in this case, goal two) better explain the observed data $\mathbf{z}_{1:t}$. This methodology is used to evaluate $w_k^{(i)} \propto p(\mathbf{z}_{1:t}^{(i)} \mid (\bar{\mathbf{g}}_m, \bar{T}_m)_k)$.

3.2.3 Sample Based Approximation of Multi-Goal Interacting Gaussian Processes

We expand the interacting Gaussian process density to take goal and waypoint duration uncertainty into account by using the motion mixture model approximation :

$$\begin{aligned} p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}) &= \frac{1}{Z}\psi(\mathbf{f}) \prod_{i=1}^{n} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) \\ &= \frac{1}{Z}\psi(\mathbf{f}) \prod_{i=1}^{n} \left(\sum_{\bar{\mathbf{g}}_m} \int_{\bar{T}_m} p(\mathbf{f}^{(i)}, \bar{\mathbf{g}}_m, \bar{T}_m \mid \mathbf{z}_{1:t}^{(i)}) \right) \\ &\approx \frac{1}{Z}\psi(\mathbf{f}) \prod_{i=1}^{n} \left(\sum_{k=1}^{N_p} w_k^{(i)} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_k, \bar{T}_k) \right). \end{aligned}$$

We wish to approximate $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ using samples. To do this, we extend the method outlined in Section 3.2.1 by adding a step to account for the Gaussian process mixture components—that is, to draw a joint sample $(\mathbf{f}^{(R)}, \mathbf{f})_l$ from the multi-goal interacting Gaussian process (mgIGP) density we first draw agent i's mixture index ζ from the discrete distribution $\{w_1^{(i)}, w_2^{(i)}, \ldots, w_N^{(i)}\}$. Given the mixture index ζ, we draw $(\mathbf{f}^{(i)})_l \sim p(\mathbf{f}^{(i)} | \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_\zeta, \bar{T}_\zeta)$. We iterate through all $N+1$ agents (including the robot), and then arrive at the joint sample weight $\eta_l = \psi((\mathbf{f}^{(R)}, \mathbf{f})_l)$. With this collection of N weights, we arrive at the approximation

$$p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}) \approx \frac{1}{\sum_{s=1}^{N} \eta_s} \sum_{l=1}^{N} \eta_l \delta([\mathbf{f}^{(R)}, \mathbf{f}]_l - [\mathbf{f}^{(R)}, \mathbf{f}]).$$

3.3 Reducing Planning to Inference

In this section, we explain how the IGP (and mgIGP) density $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ can be interpreted as a "navigation density"—that is, in our model, navigation can be understood as a statistic of prediction. We also explain how a noncooperative planner can be implemented in this manner.

3.3.1 Interacting Gaussian Processes for Navigation

Our model $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ immediately suggests a natural way to perform navigation: at time t, find the *maximum a-posteriori* (MAP) assignment for the posterior

$$(\mathbf{f}^{(R)}, \mathbf{f})^* = \arg\max_{\mathbf{f}^{(R)}, \mathbf{f}} p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t}),$$

and then take $\mathbf{f}^{(R)*}(t+1)$ as the next action in the path (where $t+1$ means the next step of the estimation). At time $t+1$, we receive a new observation of the agents and the robot, update the posterior to $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t+1})$, find the MAP assignment again and choose $\mathbf{f}^{(R)*}(t+2)$ as the next step in the path. We repeat this process until the robot has arrived at its destination.

We point out that this approach is a special case of the duality between stochastic optimal control and approximate inference discovered in Toussaint [112] and fully formalized in Rawlik et al. [85]. Our approach is illustrated in Figure 3.7.

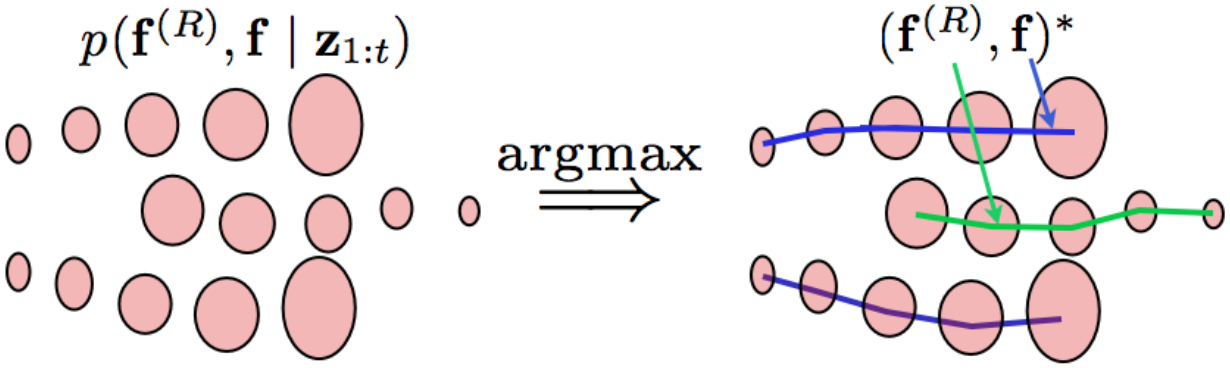


Figure 3.7: Navigation as a statistic of the prediction density $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$. Green and blue lines indicate the $\arg\max_{\mathbf{f}^{(R)}, \mathbf{f}} p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$; the green line is interpreted as navigation commands for the robot. In this way, behavioral models of crowds become navigation protocols.

3.3.2 Noncooperative Planner

We can also leverage the Gaussian process prediction models to do *noncooperative* planning. That is, we can plan through a crowd that we do not expect to respond to the robot—the robot merely maximizes the distance between itself and the expected independent trajectories of each pedestrian

while minimizing the length of the path to the goal. In Chapter 5, we test this noncooperative planner in dense human crowds.

A slight modification of the importance sampling technique detailed in Section 3.2.1 allows us to do this: for agent i, instead of drawing samples such that $(\mathbf{f}^{(i)})_l \sim p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) = GP(m_t^{(i)}, k_t^{(i)})$, we only draw one sample—the most probable sample—according to $(\mathbf{f}^{(i)})_l = m_t^{(i)}$. For the robot, we continue to draw samples according to $(\mathbf{f}^{(R)})_l \sim p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}^{(R)}) = GP(m_t^{(R)}, k_t^{(R)})$. Each joint sample $(\mathbf{f}^{(R)}, \mathbf{f})_l$ is then weighted according to the potential function $\psi((\mathbf{f}^{(R)}, \mathbf{f})_l)$, and the sample with the highest weight is chosen as the navigation command for time t. Once we receive new data $\mathbf{z}_{t+1}$, the process is repeated, and the navigation command for time $t+1$ is found. More generally, this procedure is just the sampling based approximation of

$$(\mathbf{f}^{(R)})^* = \arg\max_{\mathbf{f}^{(R)}} \psi(\mathbf{f}^{(R)}, m_t^{(1)}, \ldots, m_t^{(n)}) p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}^{(R)}). \tag{3.3}$$

Additionally, if we wish to use the Gaussian process mixture models

$$p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}) \approx \sum_{k=1}^{N_p} w_k^{(i)} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}, \bar{\mathbf{g}}_k, \bar{T}_k),$$

then we follow the same procedure: for each agent i, we choose the most probable agent trajectory and then sample the robot path with highest potential function value. The most probable trajectory for agent i is the mean of the most likely mixture component. Thus, we find the largest weight $w_k^{(i)}$, and then choose $(\mathbf{f}^{(i)})_l = m_t^{(i,k)}$, where we add the additional superscript k in $m_t^{(i,k)}$ to indicate mixture component k. By optimizing the robot's trajectory against the most probable agent predictions, we produce an algorithm that is highly similar to the planners described in Du Toit [28], Aoude et al. [5, 4] and Joseph et al. [52].

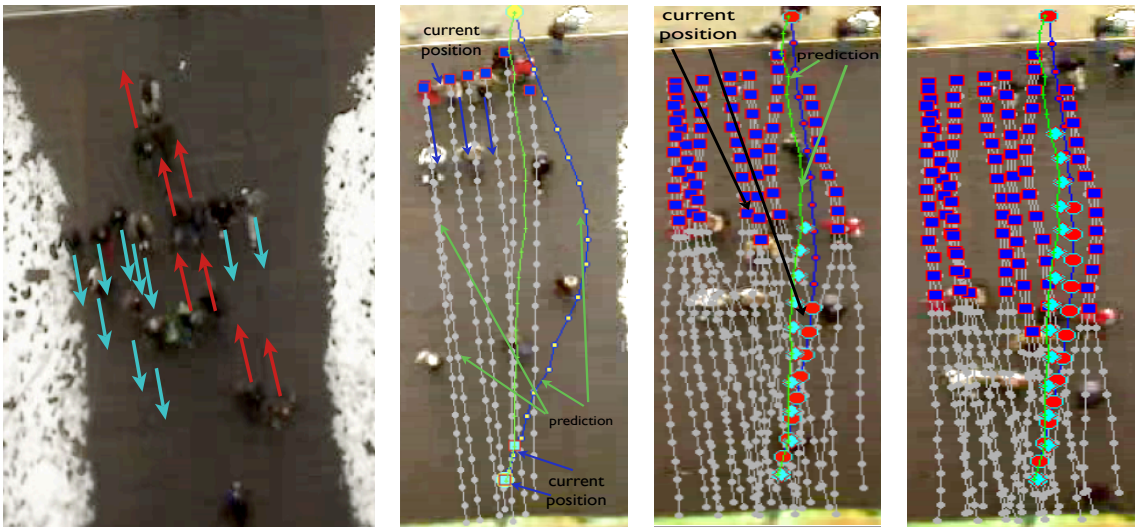
We point out that other noncooperative planners were available. For instance, we could have minimized the *expected* probability of collision using Kalman filter prediction or Gaussian process mixture model prediction. However, both of these approaches are the receding horizon control implementation of

$$\begin{aligned}\mathbf{f}_t^* &= \arg\min_{\mathbf{f}^{(R)}} J(\mathbf{f}^{(R)} | \mathbf{z}_{1:t}) \\ &= \arg\min_{\mathbf{f}^{(R)}} \int c(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)}) p(\mathbf{f} \mid \mathbf{z}_{1:t}) d\mathbf{f}.\end{aligned}$$

As shown in Section 2.4.1, both these approaches have larger objective function costs than the planner of Equation 3.3, meaning that average performance in dense crowds is guaranteed to be inferior. Accordingly, we chose to experiment with the planner of Equation 3.3.

3.4 Simulation Experiments

3.4.1 Experimental Setup: Crowded Pedestrian Data



(a) *Example pedestrian interaction* (b) *Noncooperative* (c) *IGP* (d) *IGP*

Figure 3.8: (a) Crowded still from the ETH data sequence. Near the center of the group is a subgroup of about 6 people moving upwards (red arrows) through a crowd of about 10 people moving down (cyan arrows). Simulations were run on this particular scenario, with IGP performing (in terms of safety and path length) about the same or slightly better than the actual pedestrians, and greatly outperforming state of the art methods, such as seen in [28]. (b-d) The blue squares over the gray lines are the agents traveling downward (lowest dot is current position), the cyan diamond over the green line is the pedestrian walking upwards through the crowd; IGP is red circles on top of blue prediction line, and the noncooperative planner is blue prediction line. In (b) the noncooperative planner chooses an overcautious path because the crowd is too dense. In (c) and (d), the IGP follows nearly the same path as the pedestrian in green, validating the model. This set of figures illustrates the free space created by the pedestrian walking through the crowd—this is the interaction we capture with the IGP model.

Before we instrumented Caltech's Chandler dining hall, we first evaluated the IGP approach on a data set of over 8 minutes of video recorded from above a doorway of a university building at ETH Zurich (see [79] for more details of the video collection process and how to access the data). This data set exhibits high crowd density, i.e., people frequently pass by one another fairly closely. As an example, see Figure 3.8(a) for one frame of the data sequence in which the crowds are dense. In this frame, a number of pedestrians are heading down towards the doorway (cyan arrows) while a number of other people (red arrows) head *into* and *through* the crowd.

We tested the IGP algorithm on variations of just these types of scenarios (one crowd or person intersecting another crowd); our task was to utilize the navigation density in combination with the particle filtering inference method to do navigation through these crowds.

Given the type of data that we experimented with, we now explain our performance metric: For

navigation, we are interested in two quantities: *path length* (the Euclidean path distance in $x - y$ space taken by the robot from start to finish), and *safety margin* (the nearest distance that the robot ever came to another pedestrian during a run). We hope to minimize the path length while maximizing the safety margin.

We measure both these quantities in pixel values, because transforming back to "real" distances (meters, for instance) would be too inaccurate. Importantly, we have baselines for the two metrics in pixels. For path length, we tended to see pedestrians take paths which ranged from about 350-390 pixels. For the safety margin, we often observed pedestrians within 11-12 pixels of one another, although never any closer. Based on this empirical observation of human behavior, we chose any separation distance above 13 pixels to be "safe". Furthermore, we can roughly estimate 13 pixels to be about the width of a person from shoulder to shoulder. Based on this, we chose the value of h in our potential function ψ to be 13 pixels.

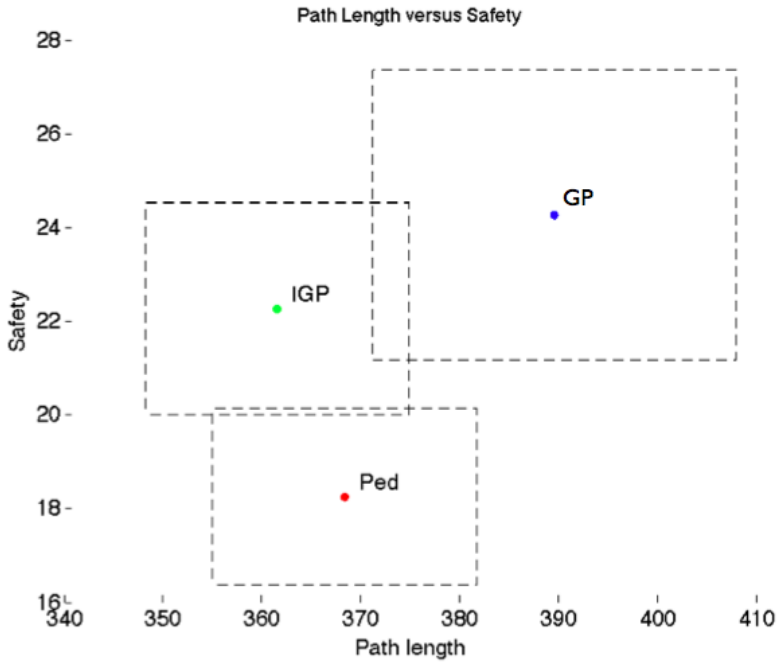


Figure 3.9: Path length versus safety over 10 runs. IGP (IGP) outperforms pedestrians in both safety and path length, while the noncooperative planner (GP) is inappropriate for this application.

As a validation of the methods developed above, we tested against a dataset of human crowds, rather than simulated dynamic agents. In order to test joint collision avoidance, we gave the IGP planner and the noncooperative planner the same start and goal states as a human navigating through a crowd, and ran the algorithms simultaneously with the human. In other words, the person created space, and we tested the algorithms to see if they would *anticipate* that space. The fact that the IGP took nearly identical paths to the humans and the noncooperative planner

chose highly conservative paths justified, to some extent, our approach. Furthermore, examinations of *planned* paths at early stages in the experiment showed the IGP expecting the opening in the crowd, while the noncooperative planner expected no such event.

3.4.2 Navigation Performance

We begin this section with anecdotal evidence of how our algorithm performs in comparison to both pedestrians and the noncooperative planner, in Figure 3.8. Note that for all 10 experiments we ran, this behavior was typical: the IGP performed similarly or better than the pedestrian, and the noncooperative planner took evasive action, usually going to the far outside to avoid the crowds.

Figure 3.9 is the main result of the simulation experiments. In Figure 3.9, we present the results of the various algorithms over 10 experiments. Each box surrounding the colored dots represents the standard error bars over the 10 experiments. The IGP (green dot) had a mean safety of around 22 pixels, with standard error ranging over 2 pixels, and mean path length of around 362, with standard error around 12. Table 3.1 presents details for the 10 individual experiments. Columns labeled s refer to safety (in pixels), and ℓ refers to path length (pixels).

Table 3.1: Navigation results: IGP (IGP) versus pedestrian versus noncooperative planner (GP). The value ℓ is the number of pixels traversed for a run (the path length in pixels), while s is the closest the agent came to any other agent during the run (also in pixels).

Run	ℓ_{ped}	s_{ped}	ℓ_{IGP}	s_{IGP}	ℓ_{GP}	s_{GP}
1	343	13	341	12	353	22
2	343	14	344	18	349	8
3	316	71	305	26	317	73
4	383	12	358	21	420	18
5	361	12	363	42	409	65
6	337	21	321	22	330	23
7	439	16	439	23	489	26
8	428	19	423	20	466	11
9	416	20	402	18	448	24
10	415	11	407	24	445	13

Figure 3.9 shows the IGP outperforming pedestrians in both safety and path length by a fairly large margin. Furthermore, the noncooperative planner is, as theoretically demonstrated earlier, inappropriate for very dense crowds—the noncooperative planner almost always takes evasive maneuvers (long path length) in an effort to avoid the crowds (large safety margin).

True validation of the IGP algorithm demanded "live" interaction, however. That is, in order to test the concept of joint collision avoidance, a robot must actually interact with human beings. This is the motivation for the content of Chapters 4 and Chapter 5.

Chapter 4

Experimental Setup

In this chapter we present the details of our experimental setup in the Chandler dining hall at Caltech. We begin by describing the actual robotic workspace in the dining hall, including details about size and the instrumentation of the workspace. Next, we provide details of our overhead pedestrian tracking system, and conclude with a discussion of the transformation of our robotic platform from a bare Evolution Robotics ER-1® to an embodied Pioneer 3-DX®.

4.1 Chandler Dining Hall at Caltech

Caltech's Chandler dining hall was chosen as the ideal location for the experiments we envisioned (see Figure 4.1). In the cafeteria, crowd densities could swell to above 1 person/m^2 during peak hours (shoulder to shoulder congestion). During off hours (before 1130am and after 1pm), patrons were still present, although not in such huge volume. Indeed, the two types of crowds presented unique challenges to our autonomous robot: during peak hours, the dense crowds required anticipation of human cooperation over short distances (as argued in Section 2.4.1 and validated in Section 5.5.1.2). In off peak hour crowds, patrons could be walking quite fast, and so the robot needed to be able to anticipate longer distance interactions (i.e., a person walking towards the robot from the other side of the cafeteria required a different kind of anticipation). Finally, the dining hall was made available during closed hours, providing an unprecedented experimental proving ground, that would allow us to progressively improve and milestone our experiment.

Additionally, the cafeteria crowds were highly *turbulent* (Figure 2.2). Whereas in many public spaces crowd movement will acquire strong flows in just a few directions, in Chandler dining hall

Figure 4.1: The robot workspace consists of a 20m^2 area surrounded by a buffet station (left), a pizza station (right), and a soda fountain (background). The straight line distance between the start and goal is 6m. This straight line runs roughly through the middle of the floor in the image.

the presence of eating stations (or goals; see Figure 3.2) produced many people crossing in front of and against each other. This turbulence provided an additional dimension to our experiment.

4.2 Description of the Robotic Workspace in Chandler Dining Hall

In this section, we describe how an appropriate workspace size was determined and why we chose overhead (stereo) sensing instead of onboard sensing. In Appendix D we provide a description of the computational infrastructure that was developed for the experimental workspace.

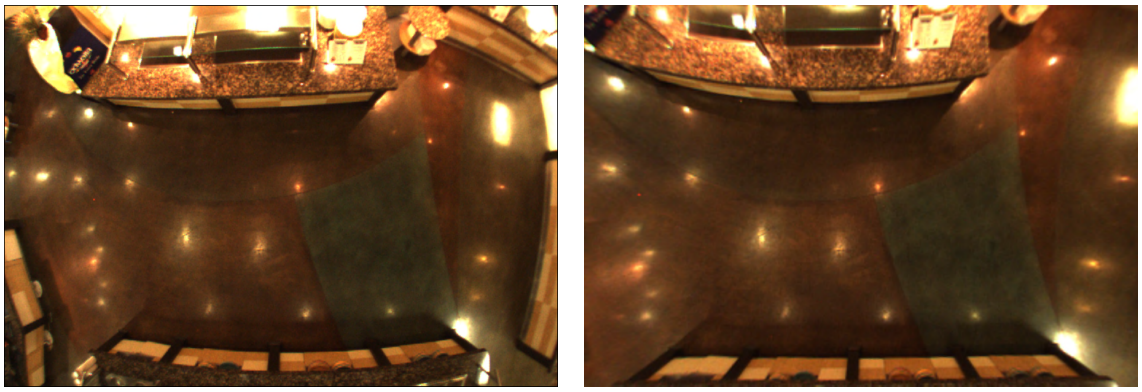
4.2.1 Instrumentation of the Workspace Versus Onboard Sensing

In dense crowds, tracking from onboard a robotic platform is a state of the art computer vision challenge. Additionally, the objective of our research was to explore navigation algorithms, not tracking and detection algorithms. For these reasons, we decided to simplify the sensing problem as much as possible. We began by choosing to instrument the cafeteria (by mounting overhead cameras in the ceiling for pedestrian tracking) rather than to instrument the robot. As it turned out, this approach had advantages all its own. Because we had a reliable tracking system that was independent of the robot, we could collect extensive crowd trajectory data logs with minimal work.

4.2.2 Overhead Instrumentation of Workspace

With the decision to instrument the workspace rather than the robot, we needed to decide what sensing modality to use, and where to place the sensors. Our experience in other robotics laboratories with overhead cameras had been largely positive. Other sensors, such as lasers or radars, were bulky, expensive, and power hungry. We thus pursued overhead camera sensing. We began with computer vision, and quickly learned that stereo vision cameras would be necessary.

4.2.2.1 Computer Vision



(a) Fish eye lens before rectification

(b) Fish eye lens after rectification

Figure 4.2: **(a)** Example image of field of view of fish eye lens *before* rectification. **(b)** Example image of field of view of fish eye lens *after* rectification. The reduction in field of view is substantial.

We began testing with a fish eye lens (an extremely wide angle field of view lens) mounted on a

Point Grey Scorpion®camera. Fish eye lenses have the tremendous advantage of an extremely large field of view. For illustration of this effect, in Figure 4.2(a) we display the unrectified fish eye view of Chandler dining hall.

The fish eye lens induces substantial warping effects on the resulting image (i.e., the image displays non-Euclidean geometries). Thus, the image must be rectified, or placed back into Euclidean coordinates, in order for any type of computations to take place. Unfortunately, this rectification results in a substantial reduction in the field of view. In Figure 4.2(b), we show the reduction in field of view that results from standard rectification procedures. Nevertheless, this field of view was still sufficiently large for the purposes of our experiments. The next step was to decide if computer vision techniques were suitable for our task.

We first tried to do tracking of pedestrians with a simple background subtraction procedure. Unfortunately, this method led to high rates of misclassification, as evidenced in Figure 4.3. It quickly

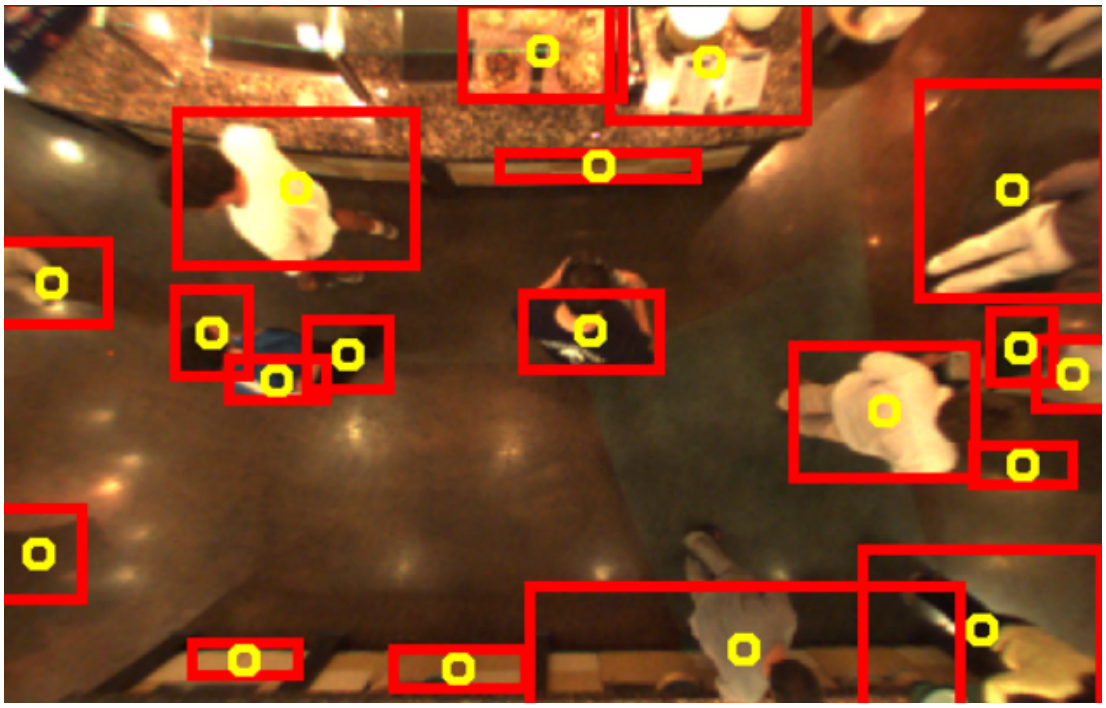


Figure 4.3: Background subtraction resulted in high rates of misclassification: the pedestrian detector picked up shadows, parts of walls, pizzas, and sometimes segmented individuals into multiple parts.

became apparent that a single camera would present extensive challenges, even with the advantage

of an overhead field of view. We thus began considering *stereo vision*, which provided the advantage of an additional range measurement (i.e., we would have access to 3-dimensional coordinates rather than 2-dimensional coordinates). We began testing with Point Grey Bumblebee®stereo cameras soon thereafter. With the discovery of existing overhead stereo vision pedestrian tracking libraries (see Section 4.3.1) we realized that this was the most appropriate sensing modality for our task.

4.3 Pedestrian Tracking System

In this section we describe the physical configuration of the stereo cameras, as well as the software used to extract the tracks of the pedestrians.

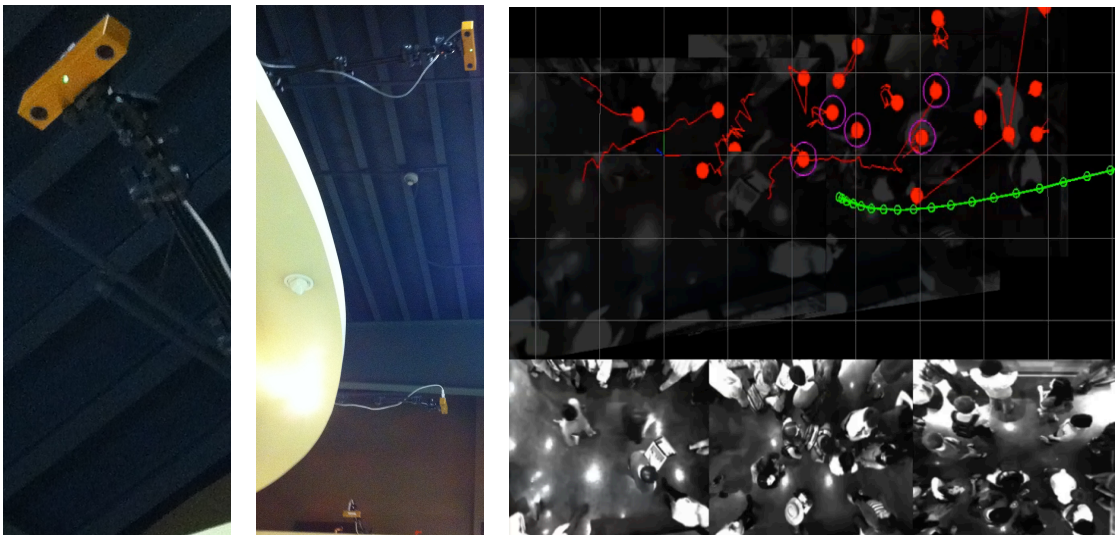
4.3.1 Overhead Stereo Vision Tracking

Our pedestrian tracking system used three Point Grey Bumblebee2 stereo cameras (Figure 4.4(a)), mounted in an overhead configuration (Figure 4.4(b)) with overlapping workspace at a nominal height of $3.5m$. The Point Grey Censys3D® software was used to provide accurate tracks of observed pedestrians at an update rate of approximately 20Hz. Censys3D® uses background subtraction with a plane fit to extract a cumulative set of 3D points belonging to pedestrains from all available stereo cameras. A clustering algorithm segments point cloud data to generate pedestrian *blobs* which are then tracked using a simple motion model with nearest neighbor data association. All identified tracks (up to a maximum of 40) were tagged with an ID and broadcast wirelessly to the Pioneer robot, which used the tracks for navigation.

Figure 4.4(c) is a screenshot of the 3D tracker used in our experiments. The bottom pane of the screenshot shows three separate overhead images from each of the stereo camera pairs (only left camera image is displayed). The top pane is our OpenGL GUI displaying all the Censys3D^{TM} tracks in red while the magenta circles indicate "important patrons" (see Section 5.2). The green path indicates the robot's current planned path. A projection of the scene is displayed underneath the tracks to provide context for the user.

Tracking the robot The tracking of the robot was done exclusively through the wheel odometry of the Pioneer 3-DX®. Since we had an overhead tracking system, robot tracking could be generalized to the crowd coordinate system by using fiducial or IR markers. For the purposes of our

experiment, however, wheel odometry was sufficient, and so an overhead robot tracking module was never developed.



(a) (b) (c)

Figure 4.4: **(a)** Stereo camera used by our pedestrian tracker. **(b)** Three stereo cameras, illustrating configuration to maximize coverage while maintaining high quality tracks. **c** Robot (wearing sun hat, bottom middle pane) navigating through densities nearing 1 person/m^2. Green dots are robot's present plan, red dots are cafeteria patrons, and magenta circles are "salient" patrons.

4.3.2 Determining Necessary Size of Workspace

In deciding the proper size of the robotic workspace, we had two competing alternatives: we needed the size of the workspace to be large enough to capture long distance interactions (to properly test the elements of the model), but small enough to maintain high fidelity pedestrian tracking (increased field of view degrades overhead camera tracking accuracy). Thus, in order to cover an extended area, multiple *synchronized and registered* cameras were necessary. Ultimately, our decision to use Point Grey's Bumblebee2® stereo camera also limited the number of cameras that we could use on a single computer. The computers available to us only had, at most, 3 PCI/PCIe boards, and each stereo camera required a dedicated PCI/PCIe slot (also, the inter-camera registration software only supported 3 cameras per instance). Synchronizing across a *second* computer would have added substantial complexity to the instrumentation, so this option was decided against.

Each Bumblebee2 stereo camera had a fixed focal length. Additionally, objects further than 3 meters from the camera had poor disparity returns, and thus poor distance information (distance information is critical for tracking). We found that, operationally, the cameras only functioned properly at a mounting height of 13 feet. At this height, given the camera's fixed focal length, the field of view of a single camera was around 3.5 meters (orthogonal to the baseline) by 5 meters (parallel with the camera baseline). Given our constraints, we chose to go with 3 stereo cameras on a single dedicated computer; we present the output of the multiple camera overlay in Figure 4.5.

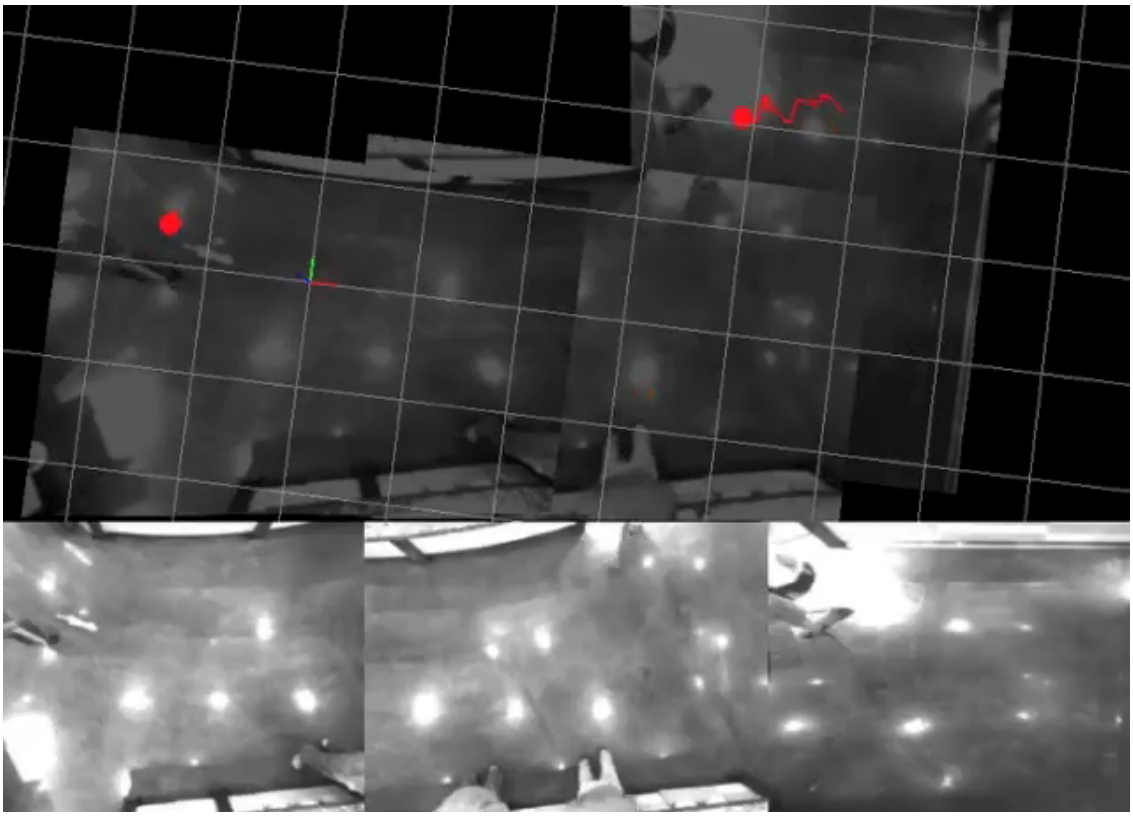


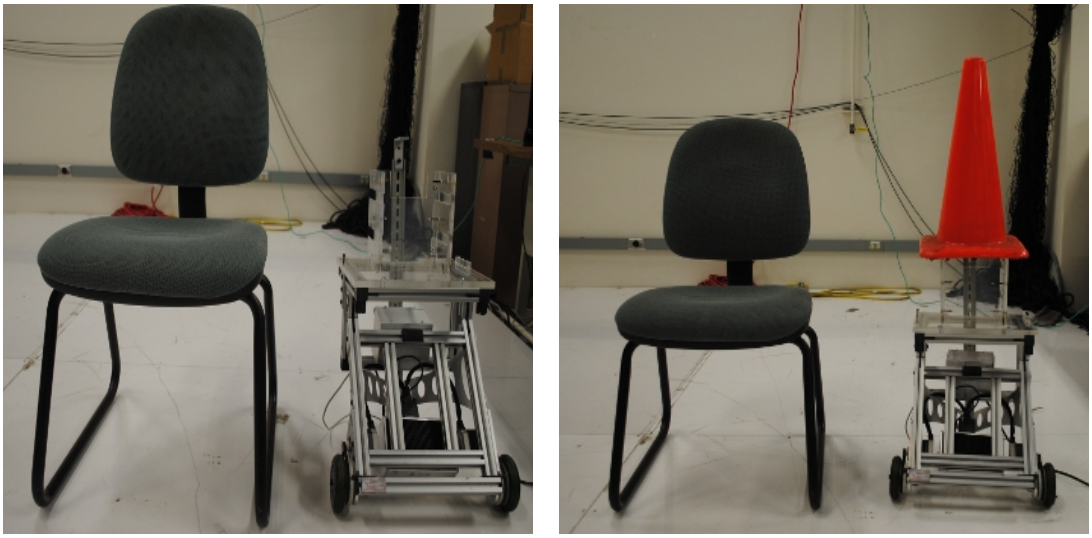
Figure 4.5: A still of the output of the 3 overhead stereo cameras. In the bottom pane, a frame from each camera is presented. In the upper pane, a frame from each camera is presented, but oriented according to their registration. The underlying grid is organized into square meter boxes. Thus, the extent of the robotic workspace is approximately 8 meters long by 3 meters wide (up to the right most camera, at which the width stretches to about 5 meters). Because of lens curvature, tracking becomes very poor near the edges of each camera's field of view. Ultimately, approximately 6 meters in length and 2.5 meters in width of reliable workspace were available for experiments.

4.4 The Robot

Before we arrived at our final robot and final form factor, we tried out several different incarnations. We discuss these various models in this section.

4.4.1 Evolution Robotics ER-1

Because of availability, we began testing with the Evolution Robotics ER-1. We quickly found that this robot was too small and not agile enough to support testing in dense crowds. Once we realized



(a) (b)

Figure 4.6: **(a)** Early version of robot: ER-1 by Evolution Robotics. **(b)** Early version of robot: ER-1 by Evolution Robotics with safety hat.

the ER-1 was inappropriate for the tests we wished to conduct, we acquired a Pioneer 3-DX, a fairly heavy duty and reliable robotic platform. In order to develop a salient but not conspicuous robot (as described in [95, 86, 82]), we began testing with the simplest robotic form factor that indicated to human observers that the robot was both sensing and comprehending its environment (see Figure 4.7(a))—a small camera atop a camera arm, with an illuminated laptop mounted on the robot's back.

(a) (b)

Figure 4.7: **(a)** The original form factor of the Pioneer 3-DX. The single arm with mounted stereo camera was essentially invisible from many vantage points. Mounted Bumblebee camera was not prominent enough to produce saliency. **(b)** The current form factor of the Pioneer 3-DX. Notice the inclusion of multiple camera arms (to produce the impression of a human like volume), an 80/20 "face" with iPad (for motion and audition purposes). The sun hat protects the robot from the bright overhead lights.

4.4.2 Pioneer 3-DX with an 80/20 Volumetric Form and iPad Face

Unfortunately, the form factor of Figure 4.7(a) was essentially invisible to cafeteria patrons, especially in crowds of density greater than .3 people/m^2. One pedestrian noted that the lack of human sized "volume" made the robot disappear into crowds. Some merely noted that the robot was too short.

As suggested in [97], we thus concentrated on filling out the volume, so that the robot had roughly the shape of a human torso. This was accomplished by mounting 3 camera arms, such that from any angle at least 2 were distinct (see Figure 4.7(b)).

Additionally, we mounted an 80/20 "head" with a computer tablet "face" at around 4 feet. We further adorned the robot's head with a sun hat. Patrons often commented that they liked the robot's appearance. At the very least, many pictures were taken of this novel cafeteria sight. The

results of our experiment (Chapter 5) support the conclusion that this form factor was adequate.

We note that, as suggested in [95], it is imperative that a robot maintain constant motion in order to be seen in crowds. Ideally, this would be accomplished by the robot maintaining some type of perpetual motion (this need not be dangerous; continuous small rotational movements are often sufficient). However, there are times when the robot is turned off; during these times, if the robot is present in the human crowd space, it is still important to maintain a presence. Furthermore, two sources of simultaneous motion (e.g., the motion of the robot base and the motion on a computer screen mounted on the robot) are typically more effective at generating human saliency than a single source. For these reasons, our motion algorithm was designed in such a way that the robot was always moving when turned on, and furthermore, an iPad face continuously played a visually salient movie.

Chapter 5

Experiments

In this chapter, we examine the navigation of a Pioneer 3-DX®nonholonomic mobile robot through dense crowds in the Chandler dining hall at Caltech. The purpose of these experiments is to understand the extent to which our model of *cooperation* (described in Sections 2.4.2 and **??**) contributes to navigation safety and efficiency in these complex and dynamic environments. To that end, we tested the following four navigation protocols: the multi-goal IGP algorithm, the single goal IGP algorithm, the noncooperative planner detailed in Section 5.3.2, and a reactive planner similar to the dynamic window approach of Fox et al. [35]. As an "upper bound" on navigation safety and efficiency, we also benchmarked human line of sight teleoperation. We emphasize that our test algorithms were chosen judiciously, in an attempt to capture the major characteristics of all existing navigation algorithms while maintaining a feasible schedule.

5.1 Testing Condition Caveats

Every effort was made to ensure that all algorithms were tested under the same conditions. As an example, the testing operator (Pete Trautman, in all cases), who was responsible for the safety of the pedestrians in the vicinity of the robot, followed the robot closely (within a few meters) during every run (in case an emergency stop was required). The close proximity of a human to the robot probably influenced the reaction of the crowd to a certain extent, and thus probably biased the performance of the robot to some extent, *for any given run.*

Unfortunately, in experiments where public safety has to be maintained, the best that can be done is to reproduce the *exact same* testing conditions for all algorithms and for all runs. We can

buttress against testing condition variation by conducting many runs, so that any undue influence on the part of the human running the experiment is equalized. For instance, the robot operator tried to maintain the same presence for every algorithm and for every run. Additionally, we collected as many runs per algorithm as was possible (approximately 3 months of testing, with nearly 500 runs collected, and around 800 attempted), so as to hopefully "wash out" any bias that may have inadvertently occurred.

In all instances, every effort was made to reproduce identical testing conditions for each algorithm and for every run.

5.2 "Important" Cafeteria Patrons

In our cafeteria experiments, we computed the 5 most "important" patrons to do prediction over. This was done so that the mgIGP planner could operate fast enough (if, in a crowd of 30 people, the planner were to do prediction over each individual, it would replan far too slowly). Doing inference over 5 people allowed the planner to operate at around 10Hz, the slowest possible replanning time for safe operation. The 5 most important patrons were taken to be the 5 patrons with the highest probability of collision with the robot; following the derivation in Du Toit and Burdick [29] we first define the *collision condition* between agent i and the robot R to be $\kappa(\mathbf{f}^{(R)}, \mathbf{f}^{(i)}) \neq \varnothing$ where κ measures the overlap (or collision) in $\mathbb{R}^2$ between two agents. The probability of collision is thus

$$P(\kappa) = \int_{\mathbf{f}^{(R)}} \int_{\mathbf{f}^{(i)}} I_\kappa(\mathbf{f}^{(R)}, \mathbf{f}^{(i)}) p(\mathbf{f}^{(R)}, \mathbf{f}^{(i)}) d\mathbf{f}^{(R)} d\mathbf{f}^{(i)}$$

where I_κ is the indicator function for whether or not a collision has occurred between $\mathbf{f}^{(R)}$ and $\mathbf{f}^{(i)}$. If we make the assumption that the robot is a disk of small area A_ϵ, then $I_\kappa(\mathbf{f}^{(R)}, \mathbf{f}^{(i)})$ restricts the integral over $\mathbf{f}^{(i)}$ to $\mathbf{f}^{(i)} \cap A_\epsilon \neq \varnothing$ so that we can approximate the probability of collision as

$$\begin{aligned} P(\kappa) &= \int_{\mathbf{f}^{(R)}} \left[\int_{\mathbf{f}^{(i)} \in A_\epsilon} p(\mathbf{f}^{(i)} \mid \mathbf{f}^{(R)}) d\mathbf{f}^{(i)} \right] p(\mathbf{f}^{(R)}) d\mathbf{f}^{(R)} \\ &\approx A_\epsilon \times \int_{\mathbf{f}^{(R)}} p(\mathbf{f}^{(i)} = \mathbf{f}^{(R)} \mid \mathbf{f}^{(R)}) p(\mathbf{f}^{(R)}) d\mathbf{f}^{(R)}. \end{aligned}$$

Additionally, we have that

$$\int_{\mathbf{f}^{(R)}} p(\mathbf{f}^{(i)} = \mathbf{f}^{(R)} \mid \mathbf{f}^{(R)}) p(\mathbf{f}^{(R)}) d\mathbf{f}^R = \int_{\mathbf{f}^{(R)}} \mathcal{N}(\mathbf{f}^{(i)}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mathcal{N}(\mathbf{f}^{(R)}; \boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R) d\mathbf{f}^R$$
$$= Z^{-1} \int_{\mathbf{f}^{(R)}} \mathcal{N}(\mathbf{f}^{(R)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{f}^R$$

(as per Appendix A.4.2) where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are combinations of $\boldsymbol{\mu}_i, \boldsymbol{\mu}_R, \boldsymbol{\Sigma}_i$ and $\boldsymbol{\Sigma}_R$ and

$$Z^{-1} = (2\pi)^{-D/2} |\boldsymbol{\Sigma}_R + \boldsymbol{\Sigma}_i|^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{\mu}_R - \boldsymbol{\mu}_i)^\top (\boldsymbol{\Sigma}_R + \boldsymbol{\Sigma}_i)^{-1} (\boldsymbol{\mu}_R - \boldsymbol{\mu}_i) \right).$$

But since the integral $\int_{\mathbf{f}^{(R)}} \mathcal{N}(\mathbf{f}^{(R)}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ evaluates to one, we have that

$$\int_{\mathbf{f}^{(R)}} p(\mathbf{f}^{(i)} = \mathbf{f}^{(R)} \mid \mathbf{f}^{(R)}) p(\mathbf{f}^{(R)}) d\mathbf{f}^R =$$
$$(2\pi)^{-D/2} |\boldsymbol{\Sigma}_R + \boldsymbol{\Sigma}_i|^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{\mu}_R - \boldsymbol{\mu}_i)^\top (\boldsymbol{\Sigma}_R + \boldsymbol{\Sigma}_i)^{-1} (\boldsymbol{\mu}_R - \boldsymbol{\mu}_i) \right).$$

Thus, we have the approximation for the probability of collision between the robot and an agent:

$$P(\kappa) \approx A_\epsilon \times (2\pi)^{-D/2} |\boldsymbol{\Sigma}_R + \boldsymbol{\Sigma}_i|^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{\mu}_R - \boldsymbol{\mu}_i)^\top (\boldsymbol{\Sigma}_R + \boldsymbol{\Sigma}_i)^{-1} (\boldsymbol{\mu}_R - \boldsymbol{\mu}_i) \right). \quad (5.1)$$

We use Equation 5.1 to determine the patrons most likely to collide with the robot.

We point out that although overhead data was available to each navigation algorithm, the 5 most important pedestrians actually used for computation were similar to those who would have been observed by an onboard sensor (for example, it was rare for an occluded pedestrian to have high probability of collision with the robot). For this reason, we anticipate that all of the algorithms (including mgIGP and IGP) would generalize well to the case of onboard sensing (assuming the onboard sensor was able to segment nearby individuals somewhat reliably).

5.3 Description of Tested Navigation Algorithms

In the first half of this section, we present details of the algorithms that we chose to test. In the second half of this section, we present an exhaustive list (so far as we are aware) of navigation algorithms that are potentially suitable for operation in human crowds. We present arguments for

why our test algorithms sufficiently represent the algorithms we decided not to test.

5.3.1 Implementation Details of mgIGP and IGP

We present the implementation details of the navigation algorithms developed in this thesis: multi-goal interacting Gaussian processes and interacting Gaussian processes.

Interacting Gaussian processes We often refer to this algorithm as the IGP planner. Implementation details of this algorithm are presented in Section 3.2.1 and in Trautman and Krause [114]. Simulation studies for this algorithm were presented in Section 3.4. As argued in Section 1.2, IGP is the first algorithm that explicitly models human cooperative collision avoidance for navigation in dense human crowds.

Multiple goal interacting Gaussian processes Using the goal model $p(\mathbf{g})$ described in Section 3.1.2.1, we implement the mgIGP as described in Sections 3.2.2, 3.2.3 and 3.3. This approach augments IGP with a Gaussian process mixture model for individual trajectory prediction.

In dense crowds, new navigation plans must be generated at around 10Hz. To accomplish this, the navigation algorithm only performed prediction over the 5 most "important" people—to the robot, the people with whom it was most likely to collide were deemed the most important.

Additionally, the mgIGP algorithm only computed the top 3 Gaussian process mixture components. As shown in Section 3.2.2 and illustrated in Figure 3.6, the mixture weight can be computed without full knowledge of the mixture component, saving substantial computational resources.

5.3.2 Baseline Algorithms

We present the navigation algorithms that mgIGP and IGP were tested against.

Noncooperative Gaussian processes: This planner proceeds in the following manner. First, given crowd data from time $t' = 1, \ldots, t$, the algorithm predicts individual trajectories using the Gaussian process mixture models of Section 3.1.2. This prediction model is similar to the state of the art crowd prediction models of Pellegrini et al. [79, 80] and Luber et al. [70]. Additionally, our mixture model is nearly identical to the state of the art prediction models used for navigation in Aoude et al. [5, 4] and Joseph et al. [52]. We also point out that when pedestrian track data indicates

linear movement, the Gaussian process mixture model predicts linear movement. Linear prediction models are common to many of the navigation algorithms that we did not test.

Second, our noncooperative planner uses importance sampling to produce a navigation command at time $t+1$ that (approximately—importance sampling is still vulnerable to finding local minima) minimizes the time to goal while maximizing safety. These two steps are iterated in a receding horizon control manner. This sampling based approximation procedure is very similar to the rapidly exploring random trees navigation method implemented in Aoude et al. [5, 4]. The presence of Gaussian process mixture models in both approaches, and the absence of cooperation modeling in both approaches, suggests a high degree of similarity between the two planning methods. Furthermore, optimizing over the most probable trajectories (rather than over distributions) is similar to the state of the art crowd navigation algorithm of Du Toit [28].

For explicit details of the implementation of the noncooperative Gaussian process planner we refer the reader to Section 5.3.2. This section also discusses other possible noncooperative planners that were available, and why we expected those algorithms to suffer the freezing robot problem.

Reactive navigation This planner moves forward in a straight line along the $x-$axis, replanning its velocity profile each time step $\Delta t \approx 0.1s$ (since the overhead tracking algorithm runs at about 10Hz, any planner in the cafeteria is limited by this constraint) so that it continues moving at the maximal speed while avoiding collision. This is accomplished in four steps.

First, the agents in the crowd are predicted forward in time approximately $0.5s$ using the Gaussian process that is not conditioned on any goals ($0.5s$ is about how long it takes the robot to come to a complete stop from maximum velocity). Next, six potential robot trajectory velocity profiles are computed (using Gaussian processes that have been conditioned on the robot's goal) along the $x-$axis. The velocity profiles range from $0m/s$ to $0.3m/s$, ($0.3m/s$ was deemed the maximum safe velocity of the robot in dense crowds), discretized in increments of $0.05m/s$. Then, each velocity profile is evaluated for potential collisions using Equation 5.1; those velocity profiles with a probability of collision above 0.3 are deemed unsafe, while those velocity profiles with a collision probability below 0.3 are considered safe (if no velocity profiles are safe, then the $0m/s$ profile is chosen). Finally, of the safe profiles, the one with the highest velocity is chosen (to maximize efficiency and safety simultaneously). This approach is motivated by the "Dynamic window approach" of Fox et al. [35].

Human teleoperation Human teleoperation was conducted at the discretion of the teleoperator, so much as was possible: we allowed the operator to maintain as much line of sight as the teleoperator considered necessary (i.e., safety was the priority). Occasionally, this meant that some operators followed the robot (e.g., some operators were more confident than others, and some operators were more confident under certain conditions).

In all, six operators teleoperated the robot, for a total of 85 runs. The data produced was low variance (as would be expected), and served as an equitable "upper bound" of dense crowd navigation performance: at all densities, the performance of the human teleoperator exceeded that of the autonomous navigation algorithm.

5.4 Description of Untested Navigation Algorithms

Unfortunately, due to time constraints, not all dynamic navigation algorithms could be tested. However, we made every effort to capture the essential characteristics of existing navigation algorithms with the algorithms we did test. In this section, we provide an overview of existing navigation approaches that we could have tested, and we provide explanation for why our test algorithms capture the essential characteristics of those algorithms.

Global dynamic window and dynamic window The Dynamic Window approach (Fox et al. [35]) was the method employed for the RHINO experiments discussed in Section 1.2. Furthermore, this method motivated our reactive planner in Section 5.3.2. We thus argue that (at least for $x-$axis movement), the Dynamic Window approach was tested. Additionally, the noncooperative planner is a generalization of this Dynamic Window approach, since it generates a (nonlinear) velocity profile based on non-parametric predictions of the dynamic agents.

The Global Dynamic Window Approach of Brock and Khatib [18] is a generalization of the Dynamic Window approach, insofar as global connectivity information is incorporated into the current motion plan, so that local minima are avoided. Since all the test algorithms we use are global planning methods, the Global Dynamic Window method is represented faithfully.

Time varying dynamic window The Time Varying Dynamic Window approach is a generalization of the Dynamic Window Approach, in which moving obstacles are modeled as moving cells in an occupancy grid map. Once again, both our reactive planner and our noncooperative planner

(that employs a general notion of prediction of dynamic agents and cost optimization) capture this model closely, at least insofar as this model is merely a cost optimization based on linear predictions.

Inevitable collision states An Inevitable Collision State (ICS, see Fraichard and Asama [37]) is a state that will result in a collision, no matter what actions the system takes. Ideally, a robot would do best to avoid such states, and many standard search algorithms, such as A^* (see Russell and Norvig [92]) can quickly find navigation strategies that do just this. However, this concept is limited to deterministic settings, and so is inapplicable in our experiments.

Probabilistic inevitable collision states Probabilistic ICS (Bautin et al. [9]) is the generalization of ICS to the case where the future trajectories of the dynamic agents are uncertain. However, this concept is merely a special case of the cost function approach introduced in Section 2.2, and thus is a special case of our noncooperative planner.

Velocity obstacles Velocity obstacles (introduced in Fiorini and Shiller [33]) were an effective early method to plan in dynamic enviroments. However, in this formulation, uncertainty was ignored. Since our cafeteria was characterized by highly noisy measurements of pedestrians, naive implementation of a velocity obstacle based planner would have been inappropriate.

Probabilistic velocity obstacles In Fulgenzi et al. [38], velocity obstacles are generalized to the case of noisy sensing (and henceforth called probabilistic velocity obstacles, or PVOs), and so are appropriate for application in human crowds. In our opinion, however, we felt that PVOs were a special case of our noncooperative planner—that is, PVOs essentially just use linear extrapolation for prediction. Thus, testing the noncooperative planner was deemed sufficient.

Reciprocal velocity obstacles The Reciprocal Velocity Obstacle (RVO) method introduced in van den Berg et al. [115] has experienced tremendous success when applied to the multi-robot coordination problem; indeed, this approach is guaranteed to be collision and oscillation free (oscillations result when two robots attempt to pass one another in the same direction. In a deterministic world, each robot corrects simultaneously, and this results in a perpetual oscillation in which neither robot makes progress). Furthermore, the algorithm has very low computational cost.

For navigation in human crowds, however, there were many complicating factors that rendered

this algorithm highly non-trivial to implement. First, and perhaps most importantly, the amount of noise in the pedestrian tracks caused the algorithm to behave erratically—sometimes, RVO seemed to not even respond to a single person walking directly at the robot. Additionally, using a Kalman Filter to "smooth" out the tracks did not improve performance. Finally, RVO makes the assumption that *all* agents are choosing velocities in a pre-specified manner. Unfortunately, humans are not so predictable. We point out that extensive attempts were made to deploy this algorithm in the Chandler dining hall. However, even under the most benign circumstances (single pedestrian), the algorithm was unsafe.

Ultimately, we decided that although this algorithm might have promise for navigation in dense crowds, the proper implementation would be highly non-trivial, due to the above issues.

RRTs with potential fields Rapidly Exploring Random Trees (RRTs) are a common and powerful method for solving navigation problems (see LaValle and Kuffner [67]). Svenstrup et al. [104] construct potential fields based on ideas from the field of proxemics (see Hall [42]); essentially, a cost map is built around each human that reflects general social preferences (for instance, humans generally prefer for dynamic agents to be in front of them), and motion prediction is accomplished with a linear extrapolation based on the current velocity. RRTs are then used to find the minimal cost robot trajectory through the potential field.

We chose not to implement this algorithm directly since it bears many resemblances to our own noncooperative planner. Indeed, our noncooperative planner places a cost around each trajectory; the primary difference is that our cost field is spherical around the obstacle, rather than ellipsoidal to reflect certain cultural inclinations.

RRTs with Gaussian process prediction In Fulgenzi et al. [40, 39], moving obstacle motion patterns are learned and represented by Gaussian processes. The planning algorithm is based on an extension of the RRT, where the likelihood of the obstacle's trajectory and the probability of collision is explicitly taken into account.

Given that the independent agent trajectory models of Fulgenzi et al. [40] are special cases of the Gaussian process mixture models developed in this thesis, and cooperation is ignored in this implementation, we felt that our noncooperative planner captured the essential contributions of this model.

The work of Aoude et al. [5, 4] and Joseph et al. [52] shares insight with the approach of

Fulgenzi et al. [40], although more sophisticated individual models are developed: motion patterns are modeled as a mixture of Gaussian processes (Rasmussen and Williams [84]) with a Dirichlet process prior over mixture weights (Teh [107]). The Dirichlet process prior allows for representation of an unknown number of motion patterns, while the Gaussian process allows for variability within a particular motion pattern. RRTs are used to find feasible paths.

The independent agent models of Aoude et al. [5, 4] were nearly identical to our Gaussian process mixture models, and they also ignored cooperation models. We thus felt that our noncooperative planner captured the essential contributions of this model.

5.5 Experimental Results: Quantitative Studies

In [97], a lengthy catalogue of metrics for determining the efficacy of a robot interacting with a human are presented. However, the authors point out that the most important metric to consider in human robot interaction experiments is *safety*; accordingly, we first evaluate the safety of the test algorithms of Section 5.3. We follow this safety study with an efficiency study. Although efficiency does not always reflect the nuanced behavior of a probabilistic algorithm interacting with humans, we felt that this study, when considered in combination with the safety study, accurately reflected the salient behaviors of our test algorithms.

Intuition for crowd density values The scale of the crowd density can be somewhat misleading since we have normalized to values between 0 and 1—bear in mind that the highest density (1 person/m^2) is a *shoulder to shoulder crowd*—see Figure 2.2. Also remember that patrons rarely stand still; this constant motion increases the complexity, confusion, and chaos of the situation. Anecdotally, the human drivers found crowd densities above 0.8 people/m^2 to be extremely difficult to teleoperate the robot through. Densities between 0.4 people/m^2 and 0.8 people/m^2 were challenging, while navigation at densities below 0.4 people/m^2 was reasonable.

5.5.1 Robot Navigational Safety in Dense Human Crowds

We define safety to be a binary variable: either the robot was able to navigate through the crowd without collision or it was not. For obvious reasons, however, we could not allow the robot to actually collide with objects (either walls or people), and so a protocol for the human monitor (Pete

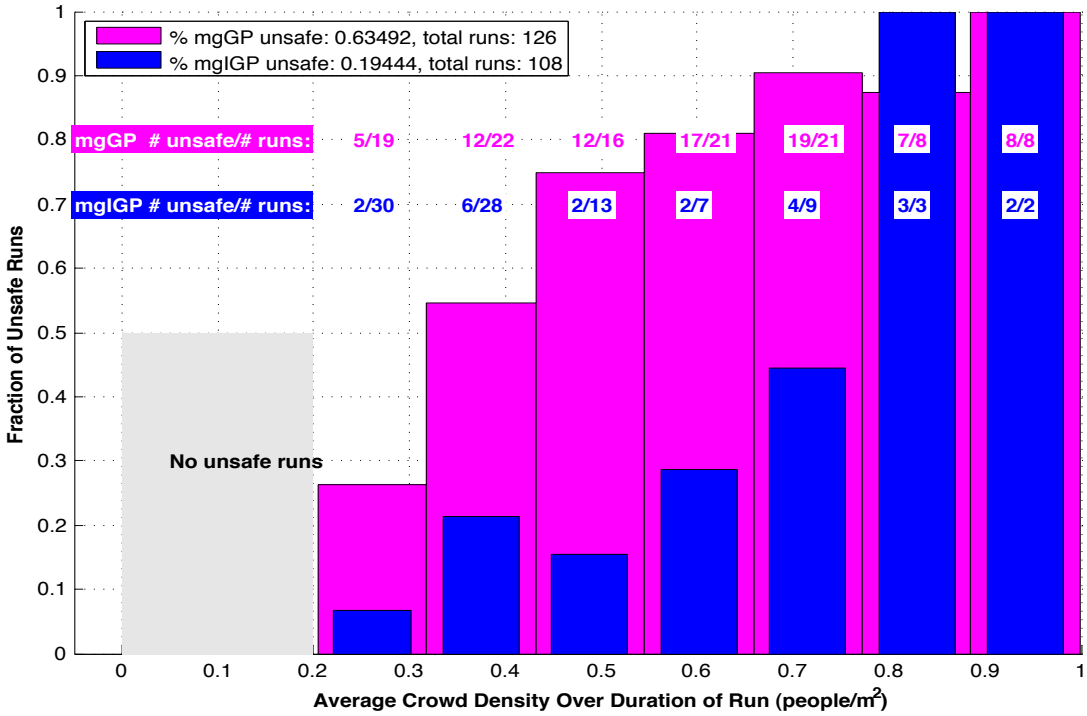


Figure 5.1: Unsafe runs for the noncooperative planner (called mgGP, in magenta) and mgIGP (in blue). Overall, the noncooperative planner fails more than 3 times as often. We also point out that at extremely high densities (above 0.8 people/m^2, when patrons are standing nearly shoulder to shoulder) all the planners consistently fail. Anecdotally, it is extremely hard to teleoperate a robot at these densities.

Trautman) was put in place: if the robot came within 1 meter of an object, and the robot did not appear to be making progress towards avoiding the collision (or, likewise, the human did not appear to be making progress towards avoiding the collision), then the robot was "emergency stopped"[1]. In other words, if the human monitor believed that a collision was imminent, then an emergency stop was required.

5.5.1.1 Noncooperative Planner versus mgIGP Planner

We first compare the safety performance of our state of the art noncooperative planner (recall Section 5.3.2) to that of the mgIGP planner in Figure 5.1. The data presented in this Figure suggests the following: modeling cooperative collision avoidance between the crowd and the robot

[1]By emergency stop, we mean that the navigation algorithm was terminated. By default, the action command immediately following termination of the navigation algorithm is the zero velocity command. Since the robot's maximum velocity is 0.3m/s, the robot is thus halted almost instantaneously.

can improve overall safety by up to a factor of $0.63/0.19 \approx 3.31$. Further inspection of Figure 5.1 reveals additional interesting structure: the safety performance of both planners degrades reliably as crowd density increases (while at densities above 0.8 people/m^2, both planners essentially cease to be safe).

We point out that the noncooperative planner is unsafe more than 50% of the time at densities as low as 0.3 people/m^2 and above. At densities of 0.55 people/m^2 and above, it is unsafe more than 80% of the time. In contrast, the interacting planner is unsafe less than 30% of the time for densities up to 0.65 people/m^2. The interacting planner is still safe more than 50% of the time at densities nearing 0.8 people/m^2, while the noncooperative planner is unsafe over 90% of the time at this high density.

We present the following explanation for the unsafe behavior of the noncooperative planner. Because the noncooperative robot essentially believes itself invisible, it has trouble finding safe paths *through* the crowd, and thus oftentimes tries to creep along the perimeter of the testing area (the testing area is bounded by walls). In our specific testing environment, this resulted in many unsafe runs: the robot's movement is simply not precise enough to avoid collisions when "wall hugging". More generally, this is a manifestation of the *freezing robot problem*, explained in Sections 2.4.1, 3.4, and illustrated in Figure 1.3. In contrast, the number of unsafe runs for the interacting planner were comparatively small because the robot was more likely to engage the crowd. By engaging the crowd, the robot elicited patron cooperation, which made navigation through the crowd safer. Additionally, by navigating in the center of the workspace, the robot was able to stay clear of hard to navigate zones, such as next to walls.

5.5.1.2 Noncooperative Planner versus IGP planner

In Figure 5.2, we compare the noncooperative planner to a "compromised" interacting planner—that is, we remove the Gaussian process mixture model individual trajectory prediction from the interacting planner (leaving it with single goal Gaussian process prediction). The noncooperative planner retains the Gaussian process mixture model prediction.

Although the results are not as stark as in Section 5.5.1.1, the IGP is still around $0.63/0.28 \approx 2.25$ times as safe as the noncooperative planner. This result suggests that for robot navigation in dense crowds, modeling cooperation is more important than high fidelity individual trajectory predictive models.

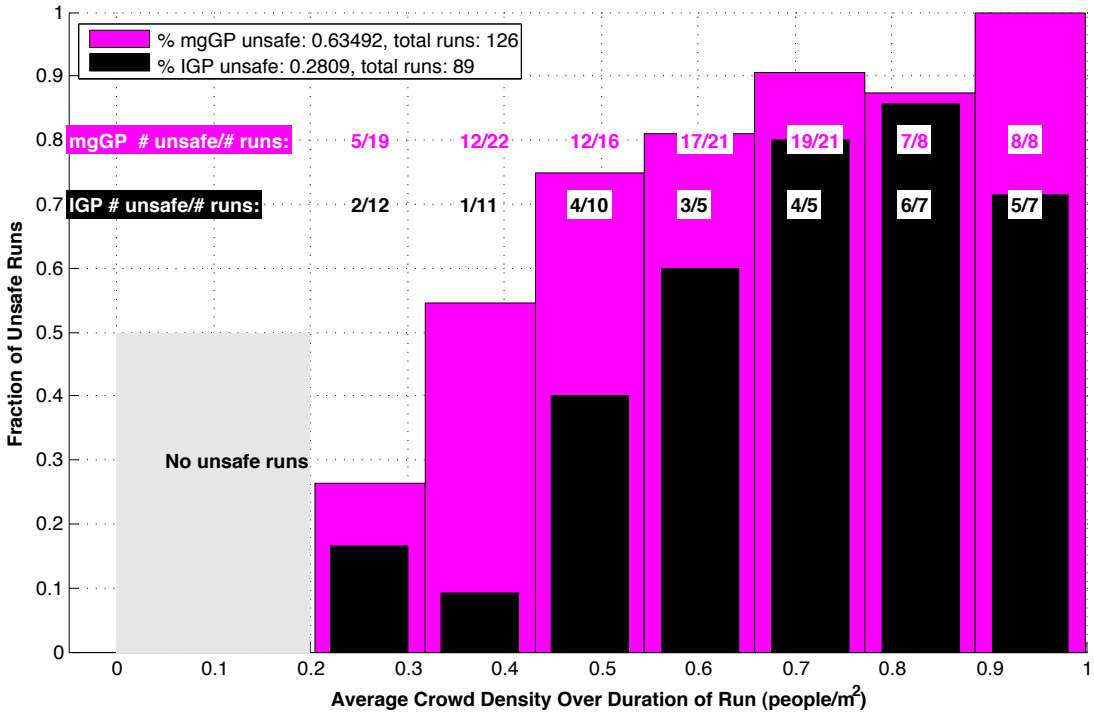


Figure 5.2: Unsafe runs for the noncooperative planner (called mgGP, in magenta) and IGP (in black). Even without goal based prediction, the interacting planner is more than twice as safe as the noncooperative planner.

5.5.1.3 Noncooperative Planner versus IGP Planner versus mgIGP Planner

We finish this Section by comparing the noncooperative planner, the mgIGP planner, and the IGP planner, in Figure 5.3. This Figure shows how each component contributes to the safety performance of the tested algorithms.

5.5.2 Robot Navigational Efficiency in Dense Human Crowds

The robot's task for every algorithm and for every run was to travel through natural, lunchtime crowds from point $A = (0, 0)$ to point $B = (6, 0)$ (in meters). This brought the robot through the center of the "filming area" in Figure D.3. Cafeteria patrons were almost entirely unscripted: they were not trained in any way, although they were warned (with signs at every entrance to the Chandler dining hall) that a robot would be present during their lunchtime routine.

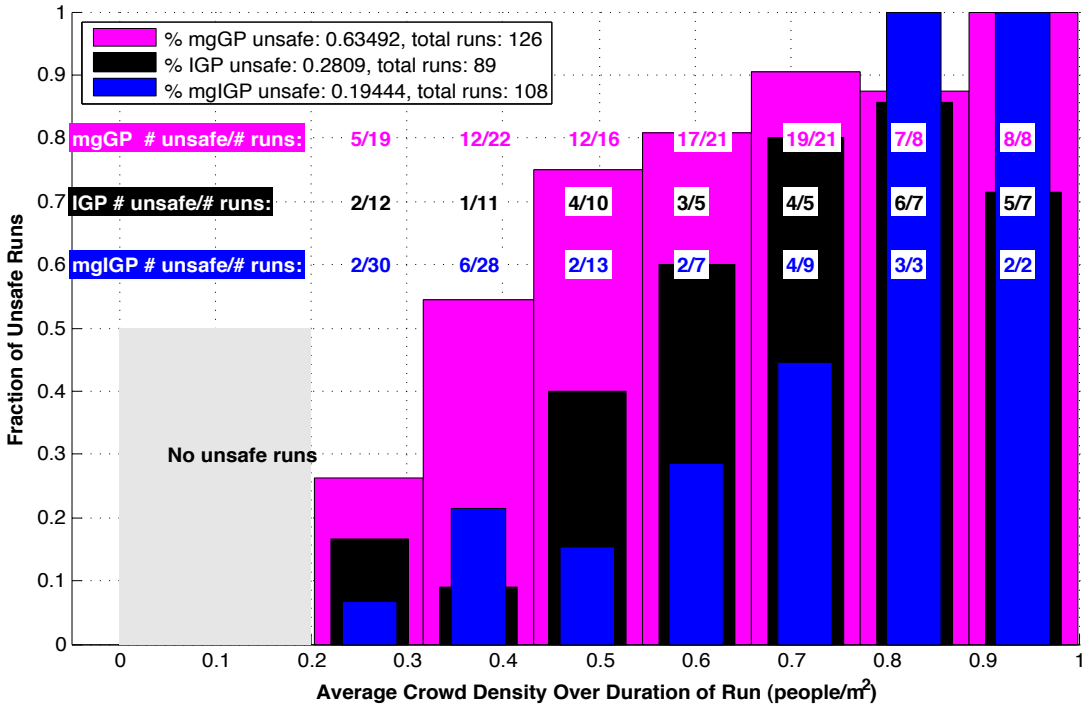


Figure 5.3: Unsafe runs for the noncooperative planner (called GP, in magenta), IGP (in blue), and mgIGP (in black).

5.5.2.1 mgIGP Planner, Noncooperative Planner, and Human Teleoperation

In Figure 5.4, we present the results of an extensive set of navigation runs (nearly 200 runs) in Chandler dining hall during lunch hours. We point out a few things. First, the number of example runs for the noncooperative planner is relatively low ($n = 40$). This is due to the typically unsafe behavior of this planner, as discussed in Section 5.5.1.1. Indeed, more runs were attempted for the noncooperative planner than for any of the other planners, precisely because the completion rate was so low. To wit, 126 runs were attempted for the noncooperative planner, 89 for the IGP planner, and 108 for the mgIGP planner.

Additionally, we point out that when the noncooperative planner did complete runs, it did so with respectable efficiency. This is easy to understand in light of the discussion of "crowd configurations" of Section 1.3. That is, the noncooperative planner was able to complete runs primarily when the crowd adopted configurations amenable to efficiency. For instance, if the patrons were standing along the perimeter of the testing space, leaving an opening through the middle, then the correct

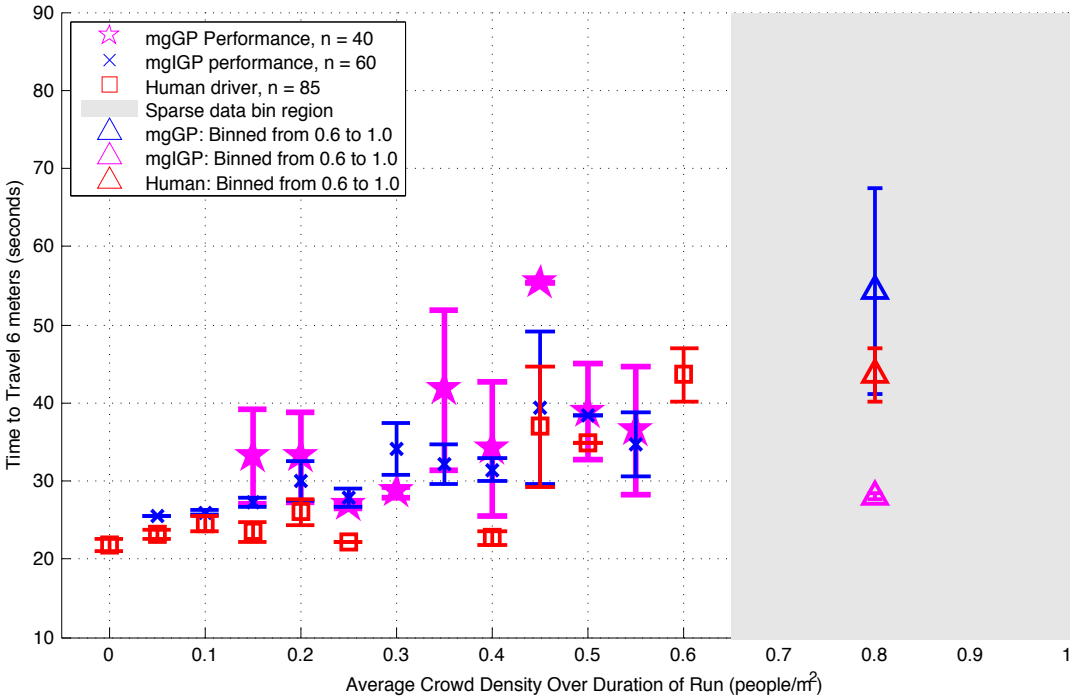


Figure 5.4: Efficiency of noncooperative planner, mgIGP, and human teleoperation.

navigation strategy did not require interaction, and so the noncooperative planner would produce an efficient run.

5.5.2.2 mgIGP Planner, Reactive Planner, and Human Teleoperation

In Figure 5.5, we present the efficiency for the reactive planner, the mgIGP planner, and human teleoperation. This figure demonstrates that, for most crowd densities, mgIGP was nearly as efficient as human teleoperation. We point out that, by definition, the human teleoperators never had to be emergency stopped—obviously, the safety of the human teleoperators was superior to any of the autonomous algorithms.

The results for the reactive planner are particularly intriguing: whereas for all the other planners (including human teleoperation) efficiency roughly increased linearly with crowd density, the reactive planner appears to grow nonlinearly with crowd density. Additionally, it is important to note that no runs for the reactive planner were collected for densities above 0.55 people/m^2. This was a result of the following: when the reactive planner started a run at a high density, it moved extremely slowly.

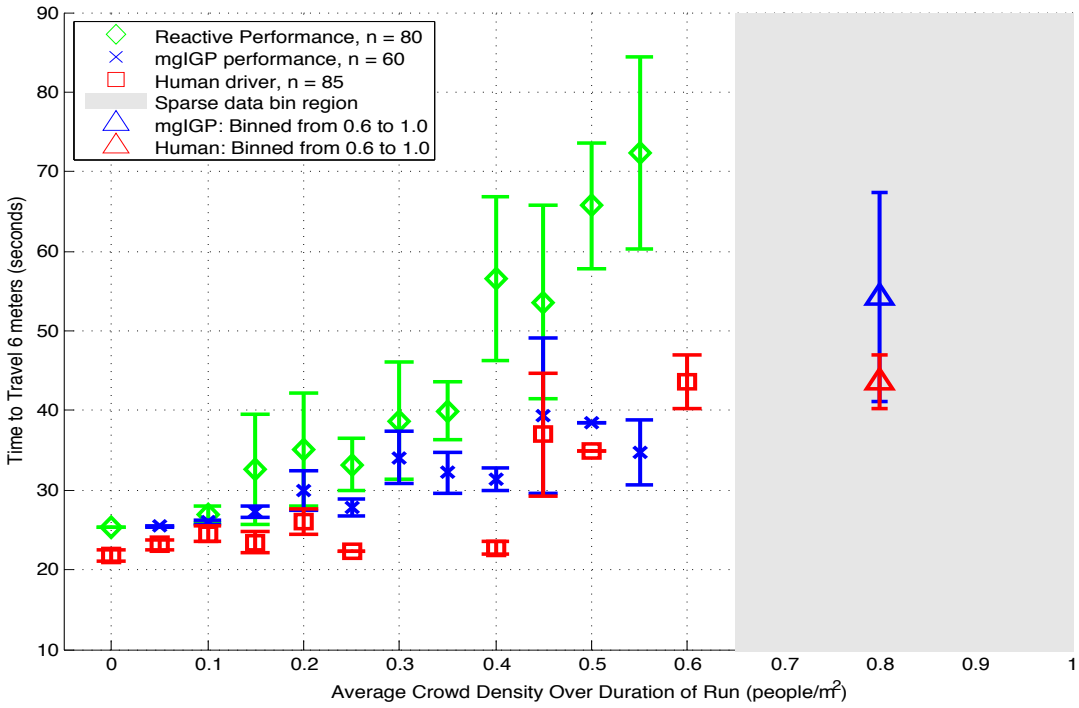


Figure 5.5: Efficiency of reactive planner, mgIGP, and human teleoperation.

Indeed, while the crowd density was above a certain amount, it almost never moved forward—the algorithm was just too cautious. So, essentially, the reactive algorithm waited until the density was low enough, and then it proceeded forward. By this time, however, the *average crowd density over the duration of the run* had dropped substantially from the *maximum* crowd density. Effectively, the reactive algorithm was unable to make progress through a crowd with an average density above 0.55 people/m^2.

5.5.2.3 IGP Planner, Noncooperative Planner, and Human Teleoperation

In Figure 5.6 we present the efficiency results for the IGP planner, the noncooperative planner, and human teleoperation. This figure provides insight into how "bare" interaction compares with a more sophisticated prediction model. Human teleoperation serves as an upper bound on efficiency.

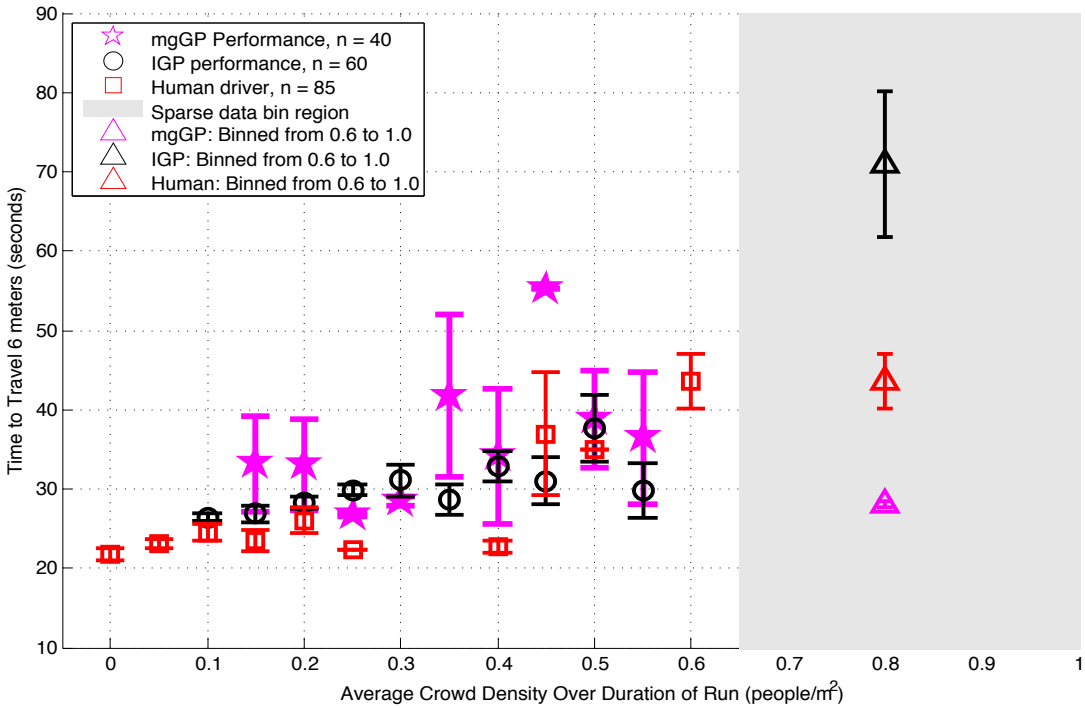


Figure 5.6: Efficiency of noncooperative planner, IGP, and human teleoperation.

5.5.2.4 mgIGP Planner, IGP Planner, and Human Teleoperation

In Figure 5.7 we present the efficiency results for the mgIGP planner, the IGP planner, and human teleoperation. This figure provides insight into how efficiency is effected when the Gaussian process mixture model of independent trajectories is removed from the interacting formulation. Human teleoperation serves as an upper bound on efficiency.

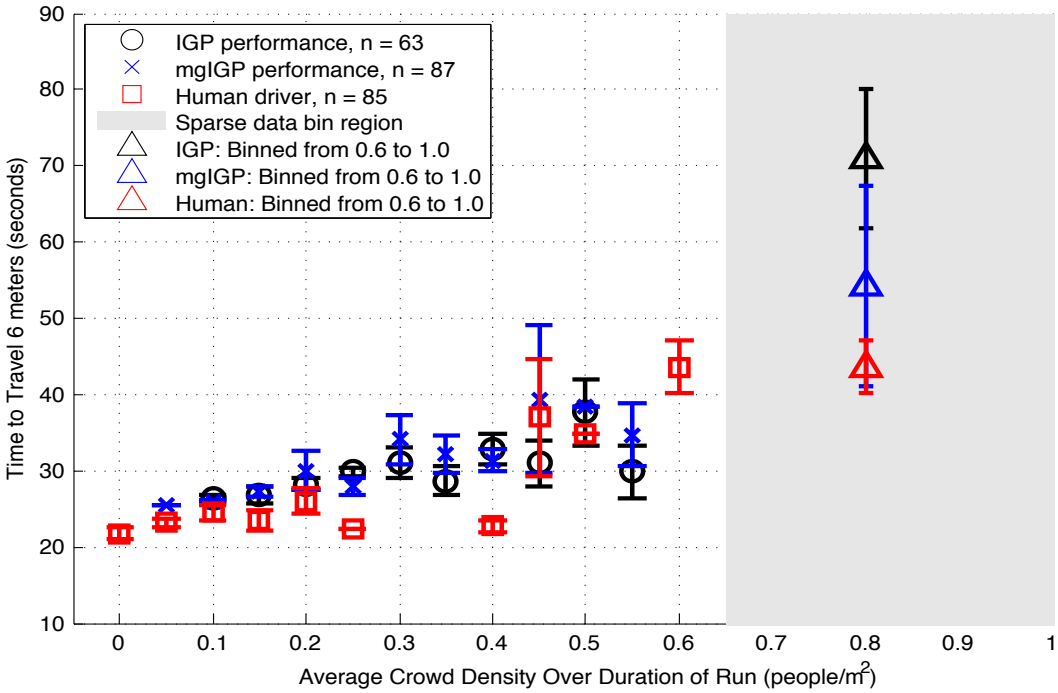


Figure 5.7: Efficiency of mgIGP, IGP, and human teleoperation. Efficiency improvement due to goal inclusion is modest.

5.6 Experimental Results: Qualitative Studies

In this section, we present qualitative details of the robot's performance. We begin the section with an image frame from a successful run of the mgIGP planner in dense crowds (Figure 5.8), and follow with a discussion of three movies, each of which illustrate various aspects of the robot's behavior in human crowds.

A highly useful behavior of the robot was that it was always in motion (see Figure 4.4(c)). This was achieved safely by doing the following: if a collision was imminent, the forward velocity was set to zero. However, the rotational velocity was not set to zero. The navigation algorithm continued generating new plans (even though the forward velocity was held at zero until collision was not imminent), and each new plan potentially pointed the robot in a new direction. Indeed, the robot was searching for a way through a challenging crowd state (see movie snippet at `http://resolver.caltech.edu/CaltechAUTHORS:20120911-130046401`).

Sometimes, this resulted in quite humorous situations: at the beginning of one run, while the

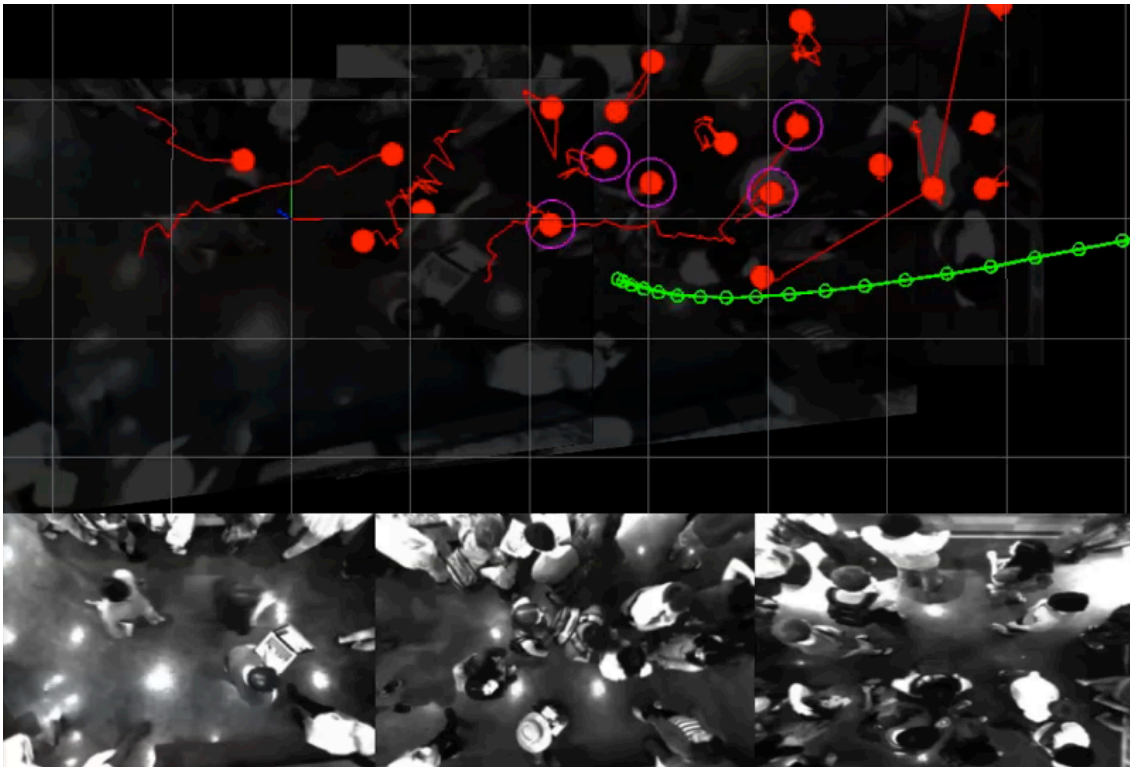


Figure 5.8: Robot (wearing sun hat, bottom middle pane) navigating through 20 people in a 10 square meter space. Robot's plan in green and people tracks in red.

navigation algorithm was still starting up, a patron approached and began inspecting the robot. The robot, sensing an imminent collision, set its velocity to zero, and began searching for a clear path (i.e., rotating in place). The patron realized what was happening, and moved along with the robot, constantly staying in front of the robot's forward velocity vector. This resulted in what we have since called the "robot dance" (see movie snippet at `http://resolver.caltech.edu/CaltechAUTHORS:20120911-125945867`).

This behavior can be quite useful in dense crowds. For instance, the reactive robot did not display this behavior—when a collision was imminent, it stopped completely. Unfortunately, a completely stopped robot is very hard for a human to understand. Is this robot turned off? Is this robot waiting for me? Meanwhile, the mgIGP robot displayed intentionality (see movie snippet at `http://resolver.caltech.edu/CaltechAUTHORS:20120911-125828298`). Animators call this behavior "readability", and it can be employed to create a more human like intelligence (see [106]).

5.7 Summary

In this chapter, we experimentally validated the mgIGP and IGP models with an empirical study of robot navigation in dense human crowds (488 runs), specifically testing how cooperation models effect navigation performance. The mgIGP algorithm performed comparably with human teleoperators in crowd densities nearing 1 person/m^2, while a state of the art noncooperative planner exhibited unsafe behavior more than 3 times as often as the multiple goal extension, and twice as often as the basic IGP approach. Furthermore, a reactive planner based on the widely used *dynamic window* approach proved insufficient for crowd densities above 0.55 people/m^2. We also showed that our noncooperative planner or our reactive planner capture the salient characteristics of nearly any dynamic navigation algorithm. Based on these experimental results and the observations of Section 2.4, we conclude that a cooperation model is critical for safe and efficient robot navigation in dense human crowds.

Chapter 6

Conclusion

6.1 Summary of Thesis Contributions

In this thesis, we considered mobile robot navigation in dense human crowds. In particular, we explored two questions. Can we design a navigation algorithm that encourages humans to cooperate with a robot? And would such cooperation improve navigation performance? We addressed the first question by developing a probabilistic predictive model of cooperative collision avoidance that we called *interacting Gaussian processes*; we then extended IGP to include multiple goals and stochastic movement duration, which we called *multi-goal interacting Gaussian processes*. We answered the second question by conducting an extensive quantitative study of robot navigation in dense human crowds (488 runs completed), specifically testing how cooperation models effect navigation performance. We found that the multi-goal interacting Gaussian processes algorithm performed comparably with human teleoperators in crowd densities near 1 person/m^2, while a state of the art noncooperative planner exhibited unsafe behavior more than 3 times as often as this multiple goal extension, and more than twice as often as the basic interacting Gaussian processes. Furthermore, a reactive planner based on the widely used "dynamic window" approach failed for crowd densities above 0.55 people/m^2.

In summary, the contribution of this thesis: based on the theoretical observations of Section 2.4, the simulation results of Section 3.4, and the experimental results of Chapter 5, we conclude that a *cooperation* model is critical for safe and efficient robot navigation in dense human crowds.

6.2 Future Work

In this section, we discuss an approach for improved approximate inference of the mgIGP distribution. We then explain how mgIGP naturally extends to a model of *shared autonomy*. We conclude by describing how this shared autonomy extension of mgIGP can solve the "assistive teleoperation" problem *exactly and efficiently*.

6.2.1 Gibbs Sampling with Metropolis-Hastings Acceptance Step for Approximate Inference of Nonlinearly Coupled Gaussian Processes

A practical challenge to robotic navigation in dense crowds is real time operation. Indeed, the robot needs to replan at around 10Hz; otherwise, *trajectory following* errors begin to accumulate. We thus consider novel inference methods for our nonlinearly coupled Gaussian process model, IGP: Gibbs sampling with a modified Metropolis-Hastings step. This approximate inference method biases the samples towards high probability regions of the distribution, potentially achieving a more efficient sampling procedure.

We first generate joint samples $[\mathbf{f}^{(R)}, \mathbf{f}]_\kappa$ according to a Gibbs sampling routine, and then accept or reject each sample according to the Metropolis-Hastings acceptance ratio γ_κ. We start by drawing the initial sample $[\mathbf{f}^{(R)}, \mathbf{f}]_0$ according to

$$\begin{aligned} \mathbf{f}_0^{(R)} &\sim p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}^{(R)}) \\ \mathbf{f}_0^{(1)} &\sim p(\mathbf{f}^{(1)} \mid \mathbf{z}_{1:t}^{(1)}) \\ &\vdots \\ \mathbf{f}_0^{(n)} &\sim p(\mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}^{(n)}) \end{aligned}$$

(note that each distribution $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)})$ can be a Gaussian process mixture of a single or of multiple

components). Once this initial sample is drawn, we draw the first sample:

$$\begin{aligned}
\mathbf{f}_1^{(R)} &\sim p(\mathbf{f}^{(R)} \mid \mathbf{f}_0^{(1)}, \mathbf{f}_0^{(2)}, \ldots, \mathbf{f}_0^{(n)}, \mathbf{z}_{1:t}^{(R)}) \\
\mathbf{f}_1^{(1)} &\sim p(\mathbf{f}^{(1)} \mid \mathbf{f}_1^{(R)}, \mathbf{f}_0^{(2)}, \ldots, \mathbf{f}_0^{(n)}, \mathbf{z}_{1:t}^{(R)}) \\
&\vdots \\
\mathbf{f}_1^{(r)} &\sim p(\mathbf{f}^{(r)} \mid \mathbf{f}_1^{(R)}, \mathbf{f}_1^{(1)}, \ldots, \mathbf{f}_1^{(r-1)}, \mathbf{f}_0^{(r+1)}, \ldots, \mathbf{f}_0^{(n)}, \mathbf{z}_{1:t}^{(R)}) \\
&\vdots \\
\mathbf{f}_1^{(n)} &\sim p(\mathbf{f}^{(n)} \mid \mathbf{f}_1^{(R)}, \mathbf{f}_1^{(1)}, \ldots, \mathbf{f}_1^{(n-1)}, \mathbf{z}_{1:t}^{(R)}).
\end{aligned}$$

The sample $[\mathbf{f}^{(R)}, \mathbf{f}]_1$ is then accepted with probability

$$\begin{aligned}
\gamma_1 &= \min\left\{1, \frac{p(\mathbf{f}_1^{(R)}, \mathbf{f}_1^{(1)}, \mathbf{f}_1^{(2)}, \ldots, \mathbf{f}_1^{(n)} \mid \mathbf{z}_{1:t})}{p(\mathbf{f}_0^{(R)}, \mathbf{f}_0^{(1)}, \mathbf{f}_0^{(2)}, \ldots, \mathbf{f}_0^{(n)} \mid \mathbf{z}_{1:t})}\right\} \\
&= \min\left\{1, \frac{p(\mathbf{f}_1^{(R)}, \mathbf{f}_1 \mid \mathbf{z}_{1:t})}{p(\mathbf{f}_0^{(R)}, \mathbf{f}_0 \mid \mathbf{z}_{1:t})}\right\}.
\end{aligned}$$

In general, we draw sample $[\mathbf{f}^{(R)}, \mathbf{f}]_\kappa$ according to

$$\begin{aligned}
\mathbf{f}_\kappa^{(R)} &\sim p(\mathbf{f}^{(R)} \mid \mathbf{f}_{\kappa-1}^{(1)}, \mathbf{f}_{\kappa-1}^{(2)}, \ldots, \mathbf{f}_{\kappa-1}^{(n)}, \mathbf{z}_{1:t}^{(R)}) \\
\mathbf{f}_\kappa^{(1)} &\sim p(\mathbf{f}^{(1)} \mid \mathbf{f}_\kappa^{(R)}, \mathbf{f}_{\kappa-1}^{(2)}, \ldots, \mathbf{f}_{\kappa-1}^{(n)}, \mathbf{z}_{1:t}^{(R)}) \\
&\vdots \\
\mathbf{f}_\kappa^{(r)} &\sim p(\mathbf{f}^{(r)} \mid \mathbf{f}_\kappa^{(R)}, \mathbf{f}_\kappa^{(1)}, \ldots, \mathbf{f}_\kappa^{(r-1)}, \mathbf{f}_{\kappa-1}^{(r+1)}, \ldots, \mathbf{f}_{\kappa-1}^{(n)}, \mathbf{z}_{1:t}^{(R)}) \\
&\vdots \\
\mathbf{f}_\kappa^{(n)} &\sim p(\mathbf{f}^{(n)} \mid \mathbf{f}_\kappa^{(R)}, \mathbf{f}_\kappa^{(1)}, \ldots, \mathbf{f}_\kappa^{(n-1)}, \mathbf{z}_{1:t}^{(R)}),
\end{aligned}$$

(we show below in Equation 6.1 that these conditional distributions are Gaussian mixtures and how

to efficiently sample from them) and sample $[\mathbf{f}^{(R)}, \mathbf{f}]_\kappa$ is then accepted with probability

$$\begin{aligned}\gamma_\kappa &= \min\left\{1, \frac{p(\mathbf{f}_\kappa^{(R)}, \mathbf{f}_\kappa^{(1)}, \mathbf{f}_\kappa^{(2)}, \ldots, \mathbf{f}_\kappa^{(n)} \mid \mathbf{z}_{1:t})}{p(\mathbf{f}_{\kappa-1}^{(R)}, \mathbf{f}_{\kappa-1}^{(1)}, \mathbf{f}_{\kappa-1}^{(2)}, \ldots, \mathbf{f}_{\kappa-1}^{(n)} \mid \mathbf{z}_{1:t})}\right\} \\ &= \min\left\{1, \frac{p(\mathbf{f}_\kappa^{(R)}, \mathbf{f}_\kappa \mid \mathbf{z}_{1:t})}{p(\mathbf{f}_{\kappa-1}^{(R)}, \mathbf{f}_{\kappa-1} \mid \mathbf{z}_{1:t})}\right\}.\end{aligned}$$

Furthermore, we note that

$$\begin{aligned}p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t}) \propto p(\mathbf{f}^{(\lambda)} \mid \mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(\lambda-1)}, \mathbf{f}^{(\lambda+1)}, \ldots, \mathbf{f}^{(n)}, \mathbf{z}_{1:t}) \times \\ p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(\lambda-1)}, \mathbf{f}^{(\lambda+1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t})\end{aligned}$$

and so

$$\begin{aligned}p(\mathbf{f}^{(\lambda)} \mid \mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(\lambda-1)}, \mathbf{f}^{(\lambda+1)}, \ldots, \mathbf{f}^{(n)}, \mathbf{z}_{1:t}) &\propto \frac{p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t})}{p(\mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(\lambda-1)}, \mathbf{f}^{(\lambda+1)}, \ldots, \mathbf{f}^{(n)} \mid \mathbf{z}_{1:t})} \\ &= \frac{\psi(\mathbf{f}^{(R)}, \mathbf{f}) \prod_{m=R}^{n} p(\mathbf{f}^{(m)} \mid \mathbf{z}_{1:t})}{\psi(\mathbf{f}^{(R)}, \mathbf{f}^{(-\lambda)}) \prod_{m\neq\lambda}^{n} p(\mathbf{f}^{(m)} \mid \mathbf{z}_{1:t})}\end{aligned}$$

where $\psi(\mathbf{f}^{(R)}, \mathbf{f}^{(-\lambda)})$ is the interaction potential function ψ over all agents *except* agent λ. Since we have that

$$\psi(\mathbf{f}^{(R)}, \mathbf{f}) = \prod_{j=i+1}^{n} \prod_{i=R}^{n} \prod_{\tau=1}^{T} \Big(1 - \alpha \exp\big(-\frac{1}{2h^2}|\mathbf{f}^{(i)}(\tau) - \mathbf{f}^{(j)}(\tau)|\big)\Big)$$

and that

$$\psi(\mathbf{f}^{(R)}, \mathbf{f}^{(-\lambda)}) = \prod_{\substack{j=i+1 \\ j\neq\lambda}}^{n} \prod_{\substack{i=R \\ i\neq\lambda}}^{n} \prod_{\tau=1}^{T} \Big(1 - \alpha \exp\big(-\frac{1}{2h^2}|\mathbf{f}^{(i)}(\tau) - \mathbf{f}^{(j)}(\tau)|\big)\Big)$$

our sampling distribution is

$$\begin{aligned}p(\mathbf{f}^{(\lambda)} \mid \mathbf{f}^{(R)}, \mathbf{f}^{(1)}, \ldots, \mathbf{f}^{(\lambda-1)}, \mathbf{f}^{(\lambda+1)}, \ldots, \mathbf{f}^{(n)}, \mathbf{z}_{1:t}) = \\ p(\mathbf{f}^{\lambda} \mid \mathbf{z}_{1:t}^{\lambda}) \prod_{i\neq\lambda}^{n} \prod_{\tau=1}^{T} \Big(1 - \alpha \exp\big(-\frac{1}{2h^2}|\mathbf{f}^{(i)}(\tau) - \mathbf{f}^{(\lambda)}(\tau)|\big)\Big).\end{aligned}$$

We point out that we are interested in

$$\begin{aligned}\mathbf{f}_{\kappa}^{(\lambda)} &\sim p(\mathbf{f}^{(\lambda)} \mid \mathbf{f}_{\kappa}^{(R)}, \mathbf{f}_{\kappa}^{(1)}, \ldots, \mathbf{f}_{\kappa}^{(\lambda-1)}, \mathbf{f}_{\kappa-1}^{(\lambda+1)}, \ldots, \mathbf{f}_{\kappa-1}^{(n)}, \mathbf{z}_{1:t}) \\ &= p(\mathbf{f}^{\lambda} \mid \mathbf{z}_{1:t}^{\lambda}) \prod_{i \neq \lambda}^{n} \prod_{\tau=1}^{T} \Big(1 - \alpha \exp\big(-\frac{1}{2h^2} |\mathbf{f}_{\lambda(\kappa)}^{(i)}(\tau) - \mathbf{f}^{(\lambda)}(\tau)|\big)\Big)\end{aligned}$$

where $\lambda(\kappa) = \kappa$ if $i < \lambda$ and $\lambda(\kappa) = \kappa - 1$ if $i > \lambda$. We thus have that

$$\mathbf{f}_{\kappa}^{(\lambda)} \sim \sum_{k=1}^{N_p} w_k^{(\lambda)} \mathcal{N}\left(\mathbf{f}^{(\lambda)} \mid \boldsymbol{\mu}_k^{\lambda}, \boldsymbol{\Sigma}_k^{\lambda}\right) \prod_{i \neq \lambda}^{n} \prod_{\tau=1}^{T} \Big(1 - \alpha \exp\big(-\frac{1}{2h^2} |\mathbf{f}_{\lambda(\kappa)}^{(i)}(\tau) - \mathbf{f}^{(\lambda)}(\tau)|\big)\Big).$$

where $\boldsymbol{\mu}_k^{\lambda}, \boldsymbol{\Sigma}_k^{\lambda}$ are the mean and covariance of N_p Gaussian process mixture components. We note that $\mathbf{f}_{\lambda(\kappa)}^{(i)}(\tau)$ is a *sample*, not a random variable, and thus each term inside of the double product can be absorbed into each Gaussian mixture component $\mathcal{N}\left(\mathbf{f}^{(\lambda)} \mid \boldsymbol{\mu}_k^{\lambda}, \boldsymbol{\Sigma}_k^{\lambda}\right)$. Accordingly, we can rewrite the sampling distribution as

$$\mathbf{f}_{\kappa}^{(\lambda)} \sim \sum_{\nu=1}^{N_{\lambda}} \tilde{w}_{\nu}^{(\lambda)} \mathcal{N}\left(\mathbf{f}^{(\lambda)} \mid \boldsymbol{\mu}_{\nu}, \boldsymbol{\Sigma}_{\nu}\right). \tag{6.1}$$

where $\tilde{w}_{\nu}^{(\lambda)}$ is $w_k^{(\lambda)}$ multiplied by the appropriate constant such that each component in this mixture remains normalized. Unfortunately, $N_{\lambda} = O(2^T 2^{n-1})$—that is, we generate a new Gaussian mixture component for each multiplicative term inside the double product. For typical values of T and n, N_{λ} is very large, and hence computing each component in the mixture is unrealistic.

However, an efficient way to rank each weight $\tilde{w}_{\nu}^{(\lambda)}$ is available. Consider the term inside the double product that consists of n_{τ} terms from the "time" product and $n_{\lambda(\kappa)}$ terms from the "agent" product. This exponential term carries with it the value $\alpha^{n_{\lambda(\kappa)}+n_{\tau}}$ that gets multiplied by the corresponding mixture weight $w_k^{(\lambda)}$. Furthermore, in order to renormalize the exponential distribution, the mixture weight $w_k^{(\lambda)}$ is also multiplied by $(\sqrt{2\pi h})^{(n_{\lambda(\kappa)}+n_{\tau})}$. That is,

$$\tilde{w}_{\nu}^{(\lambda)} = \left(\alpha\sqrt{2\pi h}\right)^{(n_{\lambda(\kappa)}+n_{\tau})} w_k^{(\lambda)}.$$

Although the value of h and α can change based on application, Figure 3.3 can provide insight about typical values (the values in this figure are based on a unit less simulation). Most importantly, we

can conclude that

$$\alpha\sqrt{2\pi h} > 1.$$

Thus for a given weight value $w_k^{(\lambda)}$ from the goal mixture (corresponding to a particular goal) we recover the largest weight $\tilde{w}_\nu^{(\lambda)}$ by including as many other agents and as many other time steps from the double product. This makes intuitive sense: the dominant terms in the mixture $\sum_{\nu=1}^{N_\lambda} \tilde{w}_\nu^{(\lambda)} \mathcal{N}\left(\mathbf{f}^{(\lambda)} \mid \boldsymbol{\mu}_\nu, \boldsymbol{\Sigma}_\nu\right)$ are those terms that have high probability goal locations for agent λ and include information from as many agents at as many time steps as possible. Since we can compute the weights $w_k^{(\lambda)}$ efficiently, we can quickly rank all of the weights $\tilde{w}_\nu^{(\lambda)}$. Based on how much computation time we have available, we can draw a mixture index $\nu_{sample} \sim \left\{\tilde{w}_\nu^{(\lambda)}\right\}_{i=1}^{N_{samples}}$. We then can use this mixture index to compute $\boldsymbol{\mu}_\nu, \boldsymbol{\Sigma}_\nu$ and thus to draw sample $\mathbf{f}_\kappa^\lambda$.

6.2.2 Shared Autonomy as an Extension of mgIGP

Current theories of *shared autonomy* are dominated by anecdotal evidence and heuristic guidelines. In Hardin and Goodrich [44] the three recognized levels of autonomy are listed: adaptive (the agent adjudicates), adjustable (the supervisor adjudicates), and mixed-initiative (the agent and supervisor "collaborate to maintain the best perceived level of autonomy"). In Fiore et al. [32], human robot collaboration schemas are organized around social, organizational and cultural factors, and in Arkin et al. [6] the role of ethological and emotional models in human-robot interaction are examined. Furthermore, actual implementations are typically designed around need, rather than principle (Murphy [78]): either the remote human operator retains complete control of the robot, or the human operator makes online decisions about the amount of autonomy the robot is given.

Importantly, the work of Dragan and Srinivasa [27] introduces principled user goal inference and prediction methods, combined with an arbitration step to balance user input and robot intelligence. However, our approach to shared autonomy as an extension of mgIGP (see Equations 6.2 and 6.3) unifies the three steps of Dragan and Srinivasa [27], thus providing a more straightforward framework in which to understand the fusion of human and machine intelligence. Most importantly, in Section 6.2.3, we show that for the special case of assistive teleoperation (the topic of Dragan and Srinivasa [27]), the mgIGP approach is *exact and efficient*.

Extending the human-robot model described in Trautman and Krause [114] provides a mathematical formulation of shared autonomy. First, recall the mgIGP model of Section 3.1.4 and the

planning and inference algorithms of Section 3.3. Next, suppose a human operator is controlling the robot from a remote location, so the robot is no longer fully autonomous (we continue the narrative of a robot navigating through a crowd $\mathbf{f}$). Rather than treating the human commands as system interrupts, we wish to understand the continuum of shared autonomy in a mathematical way. Using the navigation protocol derived using $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ as motivation, we model the joint human operator-robot *system* as

$$p(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t}) = \frac{\psi(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f})}{Z} p(\mathbf{h} \mid \mathbf{z}_{1:t}) p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) \prod_{i=1}^{n} p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)}). \tag{6.2}$$

where $\mathbf{h}$ is the is the human operator's *predicted* intentionality, modeled with a Gaussian process mixture $p(\mathbf{h} \mid \mathbf{z}_{1:t})$. The measurement data is now $\mathbf{z}_{1:t} = (\mathbf{z}_{1:t}^{\mathbf{h}}, \mathbf{z}_{1:t}^{(R)}, \mathbf{z}_{1:t}^{1}, \ldots, \mathbf{z}_{1:t}^{n})$ where $\mathbf{z}_{1:t}^{\mathbf{h}}$ are the human operator commands sent from time $1:t$. Additionally, $\psi(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f})$ is the interaction function between the human operator, robot, and human crowd. One concrete instantiation of this interaction function is

$$\psi(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f}) = \psi_{\mathbf{h}}(\mathbf{h}, \mathbf{f}^{(R)}) \psi_{\mathbf{f}}(\mathbf{f}^{(R)}, \mathbf{f})$$

where $\psi_{\mathbf{f}}(\mathbf{f}^{(R)}, \mathbf{f})$ is the cooperation function from the model $p(\mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ and $\psi_{\mathbf{h}}(\mathbf{h}, \mathbf{f}^{(R)})$ is an "attraction" model between the operator commands and the robot path. One possibility is

$$\psi_{\mathbf{h}}(\mathbf{h}, \mathbf{f}^{(R)}) = \exp\left((\mathbf{h} - \mathbf{f}^{(R)})^{\top} \boldsymbol{\Sigma}^{-1} (\mathbf{h} - \mathbf{f}^{(R)})\right)$$

so that the operator's intentionality $\mathbf{h}$ and the robot's planned path $\mathbf{f}^{(R)}$ are *merged*—this formulation of $\psi_{\mathbf{h}}(\mathbf{h}, \mathbf{f}^{(R)})$ gives high weight to paths $\mathbf{h}$ and $\mathbf{f}^{(R)}$ that are similar, while the probability of dissimilar paths decreases exponentially. Bear in mind, however, that $\psi_{\mathbf{f}}(\mathbf{f}^{(R)}, \mathbf{f})$ also gives high weight to paths $\mathbf{f}$ and $\mathbf{f}^{(R)}$ which cooperate. All of this is balanced against the (predicted) individual intentionality encoded in the Gaussian process mixtures $p(\mathbf{f}^{(i)} \mid \mathbf{z}_{1:t}^{(i)})$.

As with IGP and mgIGP, the model $p(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$ suggests a natural way to interpret shared autonomy (or shared decision making): at time t, find the MAP assignment for the posterior

$$(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f})^{*} = \arg\max_{\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f}} p(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t}), \tag{6.3}$$

and then take $(f_{t+1}^{(R)*})$ as the next robot action. As new measurements arrive, compute a new plan by

recalculating the MAP of the shared autonomy density. By choosing to interpret navigation under the model

$$p(\mathbf{h}, \mathbf{f}^{(R)}, \mathbf{f} \mid \mathbf{z}_{1:t})$$

we capture the essence of *shared autonomy* in complex environments: human commands are weighted against machine intelligence in a statistically valid way.

The key insight is that by modeling the *joint* human-robot system, we can blend human and robot capabilities in a single step to produce a superior system level decision. When the human system and the robot system are modeled independently, it becomes unclear how to fuse the complementary proficiencies of the human and robot agents.

6.2.3 Shared Autonomy with No Obstacles: an Exact Solution for Assistive Teleoperation

We discuss the special case of a single human operator tasked with teleoperating a mobile robot when no obstacles are present: $\mathbf{f} = \varnothing$. That is, we wish to do decision making with the distribution

$$\begin{aligned} p(\mathbf{h}, \mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) &= \psi(\mathbf{h}, \mathbf{f}^{(R)}) p(\mathbf{h} \mid \mathbf{z}_{1:t}) p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) \\ &= \exp\left((\mathbf{h} - \mathbf{f}^{(R)})^\top \boldsymbol{\Sigma}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}} (\mathbf{h} - \mathbf{f}^{(R)}) \right) p(\mathbf{h} \mid \mathbf{z}_{1:t}) p(\mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) \\ &= \exp\left((\mathbf{h} - \mathbf{f}^{(R)})^\top \boldsymbol{\Sigma}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}} (\mathbf{h} - \mathbf{f}^{(R)}) \right) \sum_{\mathbf{h_g}=1}^{N_\mathbf{h}} w_{\mathbf{h_g}} \mathcal{N}(\mathbf{h} \mid \boldsymbol{\mu}_{\mathbf{h_g}}, \boldsymbol{\Sigma}_{\mathbf{h_g}}) \sum_{R_\mathbf{g}=1}^{N_R} w_{R_\mathbf{g}} \mathcal{N}(\mathbf{f}^{(R)} \mid \boldsymbol{\mu}_{R_\mathbf{g}}, \boldsymbol{\Sigma}_{R_\mathbf{g}}) \end{aligned}$$

where $\boldsymbol{\mu}_{\mathbf{h_g}}$ and $\boldsymbol{\Sigma}_{\mathbf{h_g}}$ are the Gaussian process mean and covariance conditioned on the (unknown) human operator goal $\mathbf{h_g}$, and $\boldsymbol{\mu}_{R_\mathbf{g}}$ and $\boldsymbol{\Sigma}_{R_\mathbf{g}}$ are the Gaussian process mean and covariance conditioned on the robot goal $R_\mathbf{g}$. We assume that the human can choose from $N_\mathbf{h}$ discrete goals while the robot can choose from N_R discrete goals. Furthermore, we assume that the robot does not know beforehand which goal the human operator is trying to reach; in order to be successful then, the robot must not only discover which of the $N_\mathbf{h}$ goals the human wishes to reach, but also *how* (trajectory generation) the human would most desire to reach this goal. To analyze $p(\mathbf{h}, \mathbf{f}^{(R)} \mid \mathbf{z}_{1:t})$,

we introduce some quantities:

$$\bar{\mathbf{f}} = \begin{bmatrix} \mathbf{h} & \mathbf{f}^{(R)} \end{bmatrix}^\top$$

is the concatenation of the human operator trajectory and the robot trajectory, while

$$\bar{\boldsymbol{\mu}}_{\mathbf{h_g}, R_\mathbf{g}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{h_g}} & \boldsymbol{\mu}_{R_\mathbf{g}} \end{bmatrix}^\top,$$

and

$$\boldsymbol{\Sigma}_{\mathbf{h_g}, R_\mathbf{g}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{h_g}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{R_\mathbf{g}} \end{bmatrix}$$

is the block diagonal concatenation of the individual Gaussian process covariance matrices. Using this notation, we rewrite the Gaussian mixture components in a more suggestive form:

$$\mathcal{N}\left(\mathbf{h} \mid \boldsymbol{\mu}_{\mathbf{h_g}}, \boldsymbol{\Sigma}_{\mathbf{h_g}}\right) \mathcal{N}\left(\mathbf{f}^{(R)} \mid \boldsymbol{\mu}_{R_\mathbf{g}}, \boldsymbol{\Sigma}_{R_\mathbf{g}}\right) = \mathcal{N}\left(\bar{\mathbf{f}} \mid \bar{\boldsymbol{\mu}}_{\mathbf{h_g}, R_\mathbf{g}}, \boldsymbol{\Sigma}_{\mathbf{h_g}, R_\mathbf{g}}\right). \tag{6.4}$$

Furtheremore, we define the matrix

$$\Lambda = \begin{bmatrix} \mathbb{I} & -\mathbb{I} \end{bmatrix}$$

(where $\mathbb{I}$ is the identity matrix). We use this matrix when we marginalize the Gaussian process mixture components to T steps—in which case, $\mathbb{I} \in \mathbb{R}^{T \times T}$ and so $\Lambda \in \mathbb{R}^{2T \times T}$ if we are only considering one dimension (either x or y) of the trajectory. We use this matrix to rewrite our interaction function $\psi(\mathbf{h}, \mathbf{f}^{(R)})$; we begin by observing that $\Lambda \bar{\mathbf{f}} = \mathbf{h} - \mathbf{f}^{(R)}$ and so

$$\begin{aligned}
\psi(\mathbf{h}, \mathbf{f}^{(R)}) &= \exp\left((\mathbf{h} - \mathbf{f}^{(R)})^\top \boldsymbol{\Sigma}^{-1}_{\mathbf{h}, \mathbf{f}^{(R)}} (\mathbf{h} - \mathbf{f}^{(R)})\right) \\
&= \exp\left((\Lambda \bar{\mathbf{f}})^\top \boldsymbol{\Sigma}^{-1}_{\mathbf{h}, \mathbf{f}^{(R)}} \Lambda \bar{\mathbf{f}}\right) \\
&= \exp\left(\bar{\mathbf{f}}^\top (\Lambda^\top \boldsymbol{\Sigma}^{-1}_{\mathbf{h}, \mathbf{f}^{(R)}} \Lambda) \bar{\mathbf{f}}\right) \\
&= \exp\left(\bar{\mathbf{f}}^\top \bar{\boldsymbol{\Sigma}}^{-1}_{\mathbf{h}, \mathbf{f}^{(R)}} \bar{\mathbf{f}}\right) \\
&= \exp\left(\bar{\mathbf{f}} \mid \mathbf{0}, \bar{\boldsymbol{\Sigma}}^{-1}_{\mathbf{h}, \mathbf{f}^{(R)}}\right)
\end{aligned}$$

where $\bar{\mathbf{\Sigma}}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}} = \Lambda^\top \mathbf{\Sigma}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}} \Lambda$.

We thus observe that

$$p(\mathbf{h}, \mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) = \exp\left(\bar{\mathbf{f}}^\top \bar{\mathbf{\Sigma}}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}} \bar{\mathbf{f}}\right) \sum_{\mathbf{h_g}=1}^{N_\mathbf{h}} w_{\mathbf{h_g}} \mathcal{N}(\mathbf{h} \mid \boldsymbol{\mu}_{\mathbf{h_g}}, \mathbf{\Sigma}_{\mathbf{h_g}}) \sum_{R_\mathbf{g}=1}^{N_R} w_{R_\mathbf{g}} \mathcal{N}(\mathbf{f}^{(R)} \mid \boldsymbol{\mu}_{R_\mathbf{g}}, \mathbf{\Sigma}_{R_\mathbf{g}})$$

$$= \exp\left(\bar{\mathbf{f}}^\top \bar{\mathbf{\Sigma}}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}} \bar{\mathbf{f}}\right) \left[\sum_{\mathbf{h_g}=1}^{N_\mathbf{h}} w_{\mathbf{h_g}} \mathcal{N}(\mathbf{h} \mid \boldsymbol{\mu}_{\mathbf{h_g}}, \mathbf{\Sigma}_{\mathbf{h_g}}) \sum_{R_\mathbf{g}=1}^{N_R} w_{R_\mathbf{g}} \mathcal{N}(\mathbf{f}^{(R)} \mid \boldsymbol{\mu}_{R_\mathbf{g}}, \mathbf{\Sigma}_{R_\mathbf{g}})\right]$$

By Equation 6.4, we notice that all the terms inside the large brackets are of the form $\mathcal{N}\left(\bar{\mathbf{f}} \mid \bar{\boldsymbol{\mu}}_{\mathbf{h_g},R_\mathbf{g}}, \mathbf{\Sigma}_{\mathbf{h_g},R_\mathbf{g}}\right)$; multiplication by the outer exponential $\exp(\bar{\mathbf{f}} \mid \mathbf{0}, \bar{\mathbf{\Sigma}}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}})$ returns another (unnormalized) Gaussian. Thus, our distribution is a sum of Gaussians:

$$p(\mathbf{h}, \mathbf{f}^{(R)} \mid \mathbf{z}_{1:t}) = \sum_{\sigma(\mathbf{h_g}, R_\mathbf{g})} \tilde{w}_{\sigma(\mathbf{h_g},R_\mathbf{g})} \mathcal{N}\left(\bar{\mathbf{f}} \mid \bar{\boldsymbol{\mu}}_{\sigma(\mathbf{h_g},R_\mathbf{g})}, \mathbf{\Sigma}_{\sigma(\mathbf{h_g},R_\mathbf{g})}\right)$$

where $\sigma(\mathbf{h_g}, R_\mathbf{g})$ represents all the potential combinations of $(\mathbf{h_g}, R_\mathbf{g})$ resulting from the multiplication of the two mixtures and $\exp(\bar{\mathbf{f}} \mid \mathbf{0}, \bar{\mathbf{\Sigma}}^{-1}_{\mathbf{h},\mathbf{f}^{(R)}})$, and $\tilde{w}_{\sigma(\mathbf{h_g},R_\mathbf{g})}$ is $w_{\sigma(\mathbf{h_g},R_\mathbf{g})}$ after absorbing the appropriate constant normalization factors.

Critically, this means that for shared autonomy without obstacles, we can find the *robot* navigation protocol

$$(\mathbf{h}, \mathbf{f}^{(R)})^* = \arg\max_{\mathbf{h},\mathbf{f}^{(R)}} \left[p(\mathbf{h}, \mathbf{f}^{(R)} \mid \mathbf{z}_{1:t})\right]$$

exactly and efficiently—we can compute the largest weight $\tilde{w}^*_{\sigma(\mathbf{h_g},R_\mathbf{g})}$ quickly, and

$$(\mathbf{h}, \mathbf{f}^{(R)})^* = \bar{\boldsymbol{\mu}}^*_{\sigma(\mathbf{h_g},R_\mathbf{g})}$$

(thus we only have to compute the weights and a single mean). If we contrast this with recent approaches such as seen in Dragan and Srinivasa [27], we see an immediate advantage: not only do Dragan and Srinivasa [27] perform approximate inference (which is highly vulnerable to local minima since many goals are possible), but they treat goal prediction, robot trajectory generation, and arbitration between user input and robot intelligence as separate processes to be solved sequentially. Our modified mgIGP approach (along with the exact inference) allows us to solve goal prediction, trajectory generation and arbitration *simultaneously*. This simultaneity and solution optimality is

critical when the robot is trying to behave as a cooperative tool—a tool whose intent the human can understand and anticipate.

6.3 Potential Application Areas

The models developed in this thesis have application for mobile robots operating in the presence of human beings (or any intelligent agent that is not under direct control). In this section, we present applications that would immediately benefit from such a technology. Additionally, mgIGP can be understood as a model of human-robot cooperation or teaming (see Trautman [113] and the discussion in Section 6.2.2). When viewed in this way, mgIGP has important applicability beyond navigation in dense crowds.

6.3.1 Department of Defense: AFRL/Human Performance Wing

Figure 6.1: In this image, many military personnel work together to coordinate the missions of many UAVs—a classic crowd navigation problem.

In last year's *United States Air Force Report on Technology Horizons* (Dahm [23]), an executive level articulation of Air Force (AF) S&T advancements necessary to achieve mission readiness, the Chief Scientist of the AF stated that "two key areas for AF S&T investment are (i) increased use of autonomy and (ii) augmentation of human performance ... and the need to certify these high levels of adaptability and autonomy". Furthermore, conversations with AF Research Lab (AFRL) program management clarifies the motivation underlying the Chief Scientist's mandate: estimates suggest that far too many AF personnel are needed to control a single remotely piloted aircraft; simultaneously, the AF wishes to deploy as many as 10,000 drones at a time. Obviously, the current paradigm does not scale quickly enough to meet these AF goals (see Figure 6.1 for an illustration of the "crowds" in typical DoD facilities). Insofar as operating multiple drones simultaneously is a special case of the model outlined in Section 6.2.2, we suggest that the method of IGP for shared autonomy might be applicable in this scenario.

6.3.2 Industry: Boeing 737 Assembly Line

Robots have vastly improved automotive assembly line efficiency. This success is due primarily to the constrained environment of the robots. As illustrated in Figures 6.2(a) and 6.2(b), aircraft assembly lines do not have constrained environments, and so it remains unclear how one should automate these assembly lines.

Nevertheless, automation of these factories is a critical need. As with automotive assembly, aircraft assembly is composed of many discrete steps, some of which are more immediately amenable to automation than others. For instance, Boeing has identified 737 wing assembly as a point that could benefit greatly from automation. In Figure 6.2(c), a number of workers rivet a 737 wing together; Boeing's vision is to have a team of robots support the wing while humans manage the workflow and insert the rivets. The successful completion of this problem will require, at the very least, a solid methodology for robots working in crowds of humans and other robots to cooperatively accomplish a task.

6.3.3 Commercial: Telepresence Systems

A common shortcoming of modern telepresence robots is that they are *entirely* remote controlled. This leaves the robot unable to complete a basic telepresence tasks without substantial human intervention—carrying on a conversation while walking through a crowd, for instance.

(a)

(b)

(c)

Figure 6.2: **(a)** Detroit assembly line robots and a highly constrained robot operation space. **(b)** Boeing 787 assembly line. Note the lack of robots and presence of humans. **(c)** A team of workers constructing a Boeing 737 wing. Mobile robots operating in concert with the human workers could vastly improve the efficiency of this process.

In fact, the current capabilities of telepresence robots hardly enable any type of "remote presence", since the lack of onboard autonomy leaves the machines stuck in all but the most highly constrained and static environments. Building a shared autonomy layer into these robots could enable much larger operational territory. This in turn would free up the remote user to be "present" under more diverse circumstances.

Appendix A

Concepts from Probability Theory

A.1 Bayes' Theorem

Suppose that we have n (discrete or continous) random variables $x_1, \ldots x_n$. Then we call $p(x_1, \ldots x_n) = p(\mathbf{x})$ the joint probability over these variables. If we have another random variable $\mathbf{y}$ that has a dependency on $\mathbf{x}$ then the *chain rule of probability* tells us

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \mid \mathbf{y})p(\mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x}).$$

Additionally, if we wish to recover the joint over $\mathbf{x}$ and such a dependency on $\mathbf{y}$ exists, then we must use *marginalization* (see [14], [111]):

$$p\left(\mathbf{x}\right) = \int p\left(\mathbf{x}, \mathbf{y}\right)\, d\mathbf{y} = \int p\left(\mathbf{x} \mid \mathbf{y}\right) p\left(\mathbf{y}\right)\, d\mathbf{y}$$

This equation is perhaps easier to understand in the discrete case:

$$p\left(\mathbf{x}\right) = \sum_{i \in I} p\left(\mathbf{x}, \mathbf{y}_i\right) = \sum_{i \in I} p\left(\mathbf{x} \mid \mathbf{y}_i\right) p\left(\mathbf{y}_i\right)$$

We have incorporated the chain rule in the second step (both marginalizations can be useful).

These equations tell us that the probability of $\mathbf{x}$ is equal to the probability of $\mathbf{x}$ conditioned on all values of $\mathbf{y}$ times the probability of $\mathbf{y}$ itself; naturally, this invites thorny notions of recursion, but implicitly assumes that $\mathbf{x}$ is not dependent on any variables other than $\mathbf{y}$. If $\mathbf{x}$ were dependent on some other variable, we could merely subsume the new variable into $\mathbf{y}$ (think of $\mathbf{y}$ as a vector or

arbitrary length, for instance).

Bayes' Theorem is just a manipulation of the chain rule of probability:

$$p(\mathbf{x} \mid \mathbf{y})\, p(\mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x})$$

so that [7]

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y})}.$$

Bayes' theorem is deceptively simple; nevertheless, it underlies most machine intelligence machinery. Bayes' theorem can be interpreted in the following way: the probability of $\mathbf{x}$ conditioned on the variable (often an observation, or data) $\mathbf{y}$ is proportional to the likelihood value of $\mathbf{y}$ times the probability of $\mathbf{x}$ itself. The value of $p(\mathbf{x})$ leaves much to interpretation; indeed, in the Bayesian context, this is the sum total of all our prior information. It goes far beyond the standard interpretation of merely being the kinematic model of dynamic random variables.

A.2 Explanation of Bayesian Quantities

The quantity $p(\mathbf{x})$ is often called the *prior* probability of $\mathbf{x}$; this nomenclature reflects the fact that we can observe statistics governing $\mathbf{x}$ (by *prior*, we mean before collecting online measurements about $\mathbf{x}$). Indeed, if we make some observation $\mathbf{y}$ about $\mathbf{x}$, then we are no longer interested in just $p(\mathbf{x})$. Instead, we are interested now in $p(\mathbf{x} \mid \mathbf{y})$, the *posterior* probability density function (the density obtained after (*posterior* to) incorporating the most recent measurement). Bayes' theorem provides the method to transform prior distributions into posterior distributions, given likelihood functions of the data ([14]).

Notice also that given $p(\mathbf{x}, \mathbf{y})$, we can deduce any of the above conditional or marginalized probabilities. For instance, $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y})\, d\mathbf{y}$. Conditional probabilities are given by the equation $p(\mathbf{x} \mid \mathbf{y}) \propto p(\mathbf{x}, \mathbf{y})$.

A.3 Expectations

We are interested here in finding the *weighted average* of the function $f(\mathbf{x})$. For instance, the discrete expectation of $f(\mathbf{x})$ with respect to the distribution $p(\mathbf{x})$ is

$$\mathbb{E}\left[f\right] = \sum_{\mathbf{x}} f(\mathbf{x})p(\mathbf{x}).$$

This number tells us what we *expect* $f(\mathbf{x})$ to be, assuming that $\mathbf{x}$ behaves according to the distribution $p(\mathbf{x})$. Sometimes this is made explicit by using the terminology $\mathbb{E}_{p(\mathbf{x})}[f]$. For continuous $\mathbf{x}$, we have

$$\mathbb{E}_{p(\mathbf{x})}[f] = \int f(\mathbf{x})p(\mathbf{x})\, d\mathbf{x}$$

We can express marginalization as an expectation. Consider that

$$p(\mathbf{y}) = \int p(\mathbf{y}, \mathbf{x})\, d\mathbf{x} = \int p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})\, d\mathbf{x}$$

so we have

$$p(\mathbf{y}) = \mathbb{E}_{p(\mathbf{x})}[p(\mathbf{y} \mid \mathbf{x})].$$

A.4 The Gaussian Distribution

(Based on Rasmussen and Williams [84].) The multivariate Gaussian (or Normal) distribution over the D dimensional continuous random vector $\mathbf{x} = [x_1, x_2, \ldots, x_D]$ has a joint probability distribution given by

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-D/2} \mid \boldsymbol{\Sigma} \mid^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

where $\boldsymbol{\mu}$ is the mean vector of length D, and $\boldsymbol{\Sigma}$ is the (symmetric and positive definite) covariance matrix, of size $D \times D$. We sometimes write $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as a shorthand for the above.

We also point out the matrix inversion lemma

$$(Z + UWV^\top)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^\top Z^{-1}U)^{-1}V^\top Z^{-1}$$

which will be useful below in rewriting the joint distribution as a factored distribution.

Let $\mathbf{x}$ and $\mathbf{y}$ be two jointly Gaussian random vectors

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim p(\mathbf{x}, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix} \right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \tilde{A} & \tilde{C} \\ \tilde{C}^\top & \tilde{B} \end{bmatrix}^{-1} \right).$$

A.4.1 Marginals of Gaussians

The marginal distribution of $\mathbf{x}$ is $\mathcal{N}(\boldsymbol{\mu}_x, A)$, i.e.,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, A),$$

i.e.,

$$\int p(\mathbf{x}, \mathbf{y}) d\mathbf{y} = \int \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix} \right) d\mathbf{y} = \mathcal{N}(\boldsymbol{\mu}_x, A).$$

A.4.2 Products of Gaussians

The product of two Gaussians gives another (un-normalized) Gaussian:

$$\mathcal{N}(\boldsymbol{\mu}_x \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\mathcal{N}(\boldsymbol{\mu}_x \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = Z^{-1}\mathcal{N}(\boldsymbol{\mu}_x \mid \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

with

$$\boldsymbol{\Sigma}_3 = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1} \quad \text{and} \quad \boldsymbol{\mu}_3 = \boldsymbol{\Sigma}_3(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2)$$

and

$$Z^{-1} = (2\pi)^{-D/2}|\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2|^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top (\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right).$$

A.4.3 Conditionals of Gaussian Variables

The conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is

$$\mathbf{x} \mid \mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_x + CB^{-1}(\mathbf{y} - \boldsymbol{\mu}_y), A - CB^{-1}C^\top).$$

A.4.4 Generating Samples from a Gaussian Distribution

Suppose we wish to generate samples $\mathbf{x} \sim \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the mean vector of length D, and $\boldsymbol{\Sigma}$ is the (symmetric and positive definite) covariance matrix, of size $D \times D$ using only a scalar Gaussian generator (such as `randn` in Matlab) we can use the following efficient routine:

1. Compute the Cholesky decomposition L of the covariance matrix: $\boldsymbol{\Sigma} = LL^\top$. Cholesky decomposition routines are also widely available (`chol` in Matlab).

2. Generate the $D \times 1$ vector of independent scalar Gaussian samples $\mathbf{u} \sim \mathcal{N}(0, I)$.

3. Then

$$\mathbf{x} = \boldsymbol{\mu} + L\mathbf{u}$$

   has the desired statistics, i.e., $\mathbf{x} \sim \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

A.5 Sequential Bayesian Estimation

Bayesian recursion is an application of marginalization and Bayes' Rule (and various first-order Markov assumptions):

$$\begin{aligned} &\textbf{Prediction:} \quad p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t}) = \int p(\mathbf{x}_{t+1} \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) d\mathbf{x}_t \\ &\textbf{Update:} \qquad\quad p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t+1}) = \frac{p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t})}{p(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t})} \end{aligned}$$

where sometimes, the update equation is more concisely expressed as

$$p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t+1}) \propto p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t}),$$

This proportionality ignores the normalization constant

$$p(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t}) = \int p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t}) \, d\mathbf{x}_{t+1}$$

However, this normalization constant (often difficult to evaluate in practice) is very important since it makes the posterior into a true distribution, integrating to 1.

The prediction step above is really just marginalization:

$$\begin{aligned} p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t}) &= \int p(\mathbf{x}_{t+1}, \mathbf{x}_t \mid \mathbf{z}_{1:t})\, d\mathbf{x}_t \\ &= \int p(\mathbf{x}_{t+1} \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) d\mathbf{x}_t \end{aligned}$$

Alternatively, the prediction step can be viewed as the expectation of the dynamics model $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$, weighted by the *posterior* density from the previous time step, $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$:

$$p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t}) = \mathbb{E}_{p(\mathbf{x}_t \mid \mathbf{z}_{1:t})}[p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)].$$

Similarly, the update step is an application of Bayes' rule:

$$\begin{aligned} p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t+1}) &= p(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}, \mathbf{z}_{1:t}) \\ &= \frac{p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}, \mathbf{z}_{1:t}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t})}{p(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t})} \\ &= \frac{p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{z}_{1:t})}{p(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t})}. \end{aligned}$$

Appendix B

Crowd dataset

We discuss here the form of the crowd datasets collected during the experiments carried out in the student cafeteria at Caltech (Chandler cafeteria). We provide some code for extracting useful variables from the data logs.

The first shell script, `process_all.sh` references the python script `txt2mat_by_id.py`. By running `process_all.sh` on the command line, the pedestrian and robot tracks are extracted from the raw data logs and formatted as Matlab .mat files.

Once `process_all.sh` completes, open up Matlab and run `remove_dup_all.m`. This will generate the following variables which can be easily analyzed:

- `arena_tracks_ids.mat`: all the track IDs in an n×1 `int32` format.
- `arena_tracks_all.mat`, which has the following fields:

  tracks: 1×n cell, all the track data, in the same order as "IDs" in `arena_tracks_ids.mat`.
- Furthermore, each `tracks`$\{k\}$ is a struct that contains:

  `track_id`: `int32`, the track ID,

  `pos`: m×3 single, the 3D positions,

  `px`: m×2 `int32`, the pixel coordinates,

  `time_sec`: m×1 `int32`, seconds portion of time,

  `time_msec`: m×1 `int32`, millisec portion of time

The Matlab script `remove_dup_all.m` calls the scripts `load_tracks_by_id.m` and `load_tracks_by_id_nodup.m`.

```
#FILE PROCESS_ALL.SH
#!/bin/zsh

for i in *(/)
do
if [[ ! (-e $i/arena_tracks_ids.mat) ]]; then
    echo Processing $i...
    python txt2mat_by_id.py $i/arena_tracks.txt
fi
done
```

```
#FILE: TXT2MAT_BY_ID.PY

import os, sys
import re
import numpy as np
import scipy.io
import string
import commands
import time

def gen_mat_struct(track_id, path_info_list):
    time_list = [p['time_sec'][0] + p['time_msec'][0]/1000.0
                 for p in path_info_list ]

    idx_sorted = np.argsort(time_list)

    path_info_sorted = [path_info_list[i] for i in idx_sorted]

    path_pos = np.vstack(tuple([p['pos'] for p in path_info_sorted]))
    path_px = np.vstack(tuple([p['px'] for p in path_info_sorted]))
    path_time_sec = np.concatenate(tuple([p['time_sec'] for p in
path_info_sorted]))
    path_time_msec = np.concatenate(tuple([p['time_msec'] for p in
path_info_sorted]))


    return {'track_id': np.array([track_id], dtype='int'),
            'pos': path_pos,
            'px': path_px,
            'time_sec': path_time_sec,
            'time_msec': path_time_msec}



log_file = sys.argv[1]
dir_name = os.path.dirname(log_file)
mat_file_base = os.path.splitext(os.path.basename(log_file))[0]

h_log_file = open(log_file, 'r')

# count the number of lines
sts = commands.getoutput('wc -l ' + log_file)
num_tracks = int(sts.split()[0])

# track_id_array = np.zeros(num_tracks, dtype='int')
# start_time_array = np.zeros(num_tracks, dtype='float32')
```

```
# path_pos_array = np.zeros(num_tracks, dtype='object')
# path_time_array = np.zeros(num_tracks, dtype='object')

track_id_list = []
end_time_list = []
path_info_list = []

splitter = re.compile('[a-z_]+:')

i = 0
time0 = time.time()
for line in h_log_file:
    line_splitted = splitter.split(line)

    end_time = float(line_splitted[1])
    track_id = int(line_splitted[2])

    path_info = line_splitted[3].split(',')
    num_pts = len(path_info)
    path_pos = np.zeros((num_pts,3), dtype='float32')
    path_px = np.zeros((num_pts,2), dtype='int')
    path_time_sec = np.zeros(num_pts, dtype='int')
    path_time_msec = np.zeros(num_pts, dtype='int')

    for (k,path_pt) in enumerate(path_info):
        path_pt_list = path_pt.split()
        path_pos[k,:] = np.array(path_pt_list[:3], dtype='float32')
        path_px[k,:] = np.array(path_pt_list[3:5], dtype='int')
        path_time_sec[k] = int(path_pt_list[5])
        path_time_msec[k] = int(path_pt_list[6])

    track_id_list.append(track_id)
    end_time_list.append(end_time)
    path_info_list.append({'pos': path_pos,
                           'px': path_px,
                           'time_sec': path_time_sec,
                           'time_msec': path_time_msec})
    i += 1

    if i % 10000 == 1:
        print 'Processed %d out of %d (time elapsed: %g)' % (i, num_tracks,
time.time() - time0)

h_log_file.close()

scipy.io.savemat(os.path.join(dir_name,
```

```
                                mat_file_base + '_ids'),
                    {'ids': np.array(sorted(list(set(track_id_list))))})


idx_sorted = np.argsort(track_id_list)


i = 0
while i < idx_sorted.shape[0]:
   track_id_cur = track_id_list[idx_sorted[i]]
   num_occur = track_id_list.count(track_id_cur)

   mat_struct = gen_mat_struct(track_id_cur, [path_info_list[k] for k in
idx_sorted[i:i+num_occur]])
   scipy.io.savemat(os.path.join(dir_name,
                                 mat_file_base + '_%04d' % (track_id_cur)),
                    mat_struct)
   i += num_occur
   # print 'track_id: %d' % (track_id_cur)
```

```
%%%%FUNCTION LOAD_TRACKS_BY_ID
function [tracks, ids] = load_tracks_by_id(dirname)

% load the ids
id_filename = [dirname, '/', 'arena_tracks_ids.mat'];
data = load(id_filename);    % this will load variable 'ids'
ids = data.ids;

tracks = cell([1,numel(ids)]);

% load the tracks
k = 1;
for i = 1:numel(ids)
    cur_id = ids(i);
    track_filename = [dirname, '/', sprintf('arena_tracks_%04d.mat',
cur_id)];
    data = load(track_filename);
    assert(data.track_id == cur_id);
    tracks{k}.track_id = data.track_id;
    tracks{k}.pos = data.pos;
    tracks{k}.px = data.px;
    tracks{k}.time_sec = data.time_sec;
    tracks{k}.time_msec = data.time_msec;
    k = k+1;
end

%%%%FUNCTION LOAD_TRACKS_BY_ID_NODUP

function [tracks, ids] = load_tracks_by_id_nodup(dirname)

[tracks, ids] = load_tracks_by_id(dirname);

for k = 1:numel(tracks)
    track_k = tracks{k};
    time_offset = min(track_k.time_sec);
    path_time = double(track_k.time_sec - time_offset) ...
                + double(track_k.time_msec)/1000.0;
    [~, idx, ~] = unique(path_time);

    tracks{k}.pos = track_k.pos(idx,:);
    tracks{k}.px = track_k.px(idx,:);
    tracks{k}.time_sec = track_k.time_sec(idx);
    tracks{k}.time_msec = track_k.time_msec(idx);
end
```

Appendix C

Institutional Review Board Application Form: *Human Crowd Navigation*

C.1 Institutional Review Board Approval

Once we decided on the setup described in Section 4.1, Institutional Review Board (IRB) approval had to be attained. The primary concerns of the IRB were anonymity and safety. Since we would be using overhead cameras for sensing, the burden was on us to safeguard the data against privacy violation. We accomplished this using two layers of precaution: first, the data collected was low resolution and overhead (so faces were only extremely rarely present, and if they were, they were at low resolution). Second, the data collected was stored on a Caltech server with standard password protection. Also, signs were placed on each entrance to the cafeteria declaring that filming was occurring.

To guarantee safety, we did numerous things. First, the robot had a maximum velocity of $0.3m/s$—i.e., *very slow*. Next, any extending surfaces on the robot were covered with a soft guard of some sort. Most importantly, Pete Trautman constantly attended the robot, and a kill switch was always operational. Zero injuries resulted from this experiment.

We provide the original IRB documentation in this appendix.

CALIFORNIA INSTITUTE OF TECHNOLOGY
Committee for the Protection of Human Subjects
APPLICATION FOR REVIEW OF RESEARCH PROJECTS
USING HUMAN SUBJECTS

Title of Project: Human Crowd Navigation

Funding Agency: Boeing and Air Force Office of Scientific Research Multidisciplinary Research Initiative (AFOSR MURI)

Funding Identification No.: Boeing: CT-BA-GTA-1, AFOSR MURI: FA9550-06-1-0303

Original Grant Date: Boeing: May 2006-April 2011, AFOSR MURI: Jan 2009-Dec 2010

Assurance Training Certificate, attached (all personnel) Date of completion: 04/21/2010

Certification Number: 435277

1. I have read, and will comply with, the General Institutional Assurance of the California Institute of Technology.
2. Attached are application materials prepared in accordance with the instructions for Application or Review.
3. On the basis of the research proposed, I believe that the use of human subjects is:
   - Exempt. Cite the paragraph in Section .101 (p. 28012) of the Code of Federal Regulations on the Protection of Human Subjects that is the basis for the claim exemption: Section 46.101.b.2
   - Respond to items 1-3 of the Instructions and include a brief justification for claiming exemptions. In application package.
   - Non-exempt. Consent form attached or reasons for not attaching consent form:
4. Any untoward experience, emergent problem or significant deviation from my proposal will be reported to the Committee for reconsideration

Signature

Richard Murray
Name – Please print

Thomas E. and Doris Everhart Professor of Control and Dynamical Systems and Bioengineering
Title
21 April 2010
Date

Human Crowd Navigation

*PI: Richard Murray
Thomas E. and Doris Everhart
Professor of Control and Dynamical Systems
and Bioengineering

†Co-PI: Peter Trautman
Graduate Student
Control and Dynamical Systems

October 26, 2010

1 The Application Form

Included with package.

2 Human subjects involvment

2.1 How humans participate in the experiment: filming

In this experiment, anonymous human subjects are filmed from above in Chandler Cafeteria on the campus of the California Institute of Technology, during and after normal lunch time hours, in between the Pizza Byte and the buffet station (see Figure 1). The camera is very wide angle (up to 90° field of view), low resolution (300x225, see figure 2), and is mounted 12 feet above people's heads. Because of the wide angle field of view, low resolution of the camera, the distance of the camera from the people (see figure 3), and the fact that only the tops of people's heads are visible (see figure 4), the people will not be individually identifiable.

Furthermore, the majority of the original video data will be discarded, and only the locations of the people will be saved–that is, most of the saved data will only be points indicating the people's moment to moment location. Additionally, any data that is stored will be on a personal computer only accessible by Pete Trautman.

*Email: murray@cds.caltech.edu, Phone: (626) 395-6460
†Email: trautman@cds.caltech.edu, Phone: (937) 572-0468

2.2 How humans participate in the experiment: robot interaction

Once the video capture process has been finalized, we plan to eventually introduce a small robot (dimensions: 2 feet tall by 1 foot deep by 1 foot wide, see Figure 6), which is capable of a maximum speed of approximately 2 miles per hour, into the filming area. The "mission" of this robot will be to navigate through unpredictable crowds, as a service robotics demonstration.

Before any experiments are conducted in Chandler, however, we will test this robot in room 12 of Steele (which will serve as a development laboratory), with 1 person (Pete Trautman), and then increase the number of people and complexity of movement gradually. In this setting we will also test out the remote kill switch on the robot, which enables Pete Trautman, at the touch of a button, to stop the robot.

Once the robot has demonstrated sufficiently safe behavior in room 12 of Steele (collision free trajectories 100% of the time, over 20 hours of testing, with up to 3 people moving in the same environment), we will graduate the robot to Chandler cafeteria.

We will run initial Chandler experiments in the late afternoon when only a few patrons are present, in order to verify the safety of the robot in the new environment when few people are present.

We will begin testing in Chandler during off-peak hours, around 3 p.m. (Chandler cafeteria closes at 3:30 p.m.), when there are only a few people milling around. After we have demonstrated that the robot is sufficiently safe during this time period (again, 100% success rate, for 20 hours of testing), we will graduate again, up to a time period when the average crowd is approximately twice as large. We will continue this "test if robot is sufficiently safe, then graduate" cycle until we reach near peak hours.

Because these videos are observations of public behavior, and the human subjects cannot by identified in any way, and the risk of collision with the robot is extremely small we suggest that this experiment is exempt under 46.101.b.2 (see 2.3 for further analysis).

The people will not experience anything during the filming. There will be signs posted saying that filming is in progress (see figure 5), and when we begin to experiment with the robot, we will also place an additional sign warning of the robot's presence (see figure 8).

Pete Trautman has received the consent of the manager of campus dining operations, Jaime Reyes (reyes@caltech.edu), and the Assistant Vice President of Student Affairs, Housing and Dining, Peter Daily (pdaily@caltech.edu), to mount the cameras.

2.3 Risk/benefit analysis

There is no risk to the individuals in this experiment from the filming.

There is an extremely small risk of tripping/collision with the robot, although these risks will be mitigated through the use of a "robot safety hat" (an orange cone atop the robot, see figure 7), extensive safety testing, continuous monitoring within 10 feet of the robot (issuing alerts to people who come too close), and the ability to "kill" the robot's forward motion remotely.

The benefits of such an experiment are substantial; there is little to no existing data on crowd-robot interaction. Furthermore, there are only primitive implementations of robotic navigation in crowds. If successful, this experiment would have a large impact on the service robotics community.

In conclusion, the risks of this experiment are minimal, while the potential benefits are substantial.

2.4 Consent forms

Unneeded if project is deemed exempt.

2.5 Other institutions/organizations involved in research

No other institutions or organizations are involved in this research.

3 The Research Proposal

3.1 Filming

In this experiment, anonymous human subjects are filmed from above in Chandler Cafeteria on the campus of the California Institute of Technology, during and after normal lunch time hours. The camera is very wide angle (up to 90° field of view), low resolution, and is mounted 12 feet above people's heads. Because of the wide angle field of view, low resolution of the camera, the distance of the camera from the people, and the fact that only the tops of people's heads are visible, the people will not be individually identifiable.

Furthermore, the majority of the original video data will be discarded, and only the locations of the people will be saved–that is, most of the saved data will only be points indicating the people's moment to moment location. Additionally, any data that is stored will be on a personal computer only accessible by Pete Trautman.

The people will not experience anything during the filming. There will be signs posted saying that filming is in progress, but the filming will in no way affect the people.

3.2 Robot interaction

Once the video capture process has been finalized, we plan to eventually introduce a small robot (dimensions: 2 feet tall by 1 foot deep by 1 foot wide, see Figure 6), which is capable of a maximum speed of approximately 2 miles per hour, into the filming area. The "mission" of this robot will be to navigate through unpredictable crowds, as a service robotics demonstration.

Before any experiments are conducted in Chandler, however, we will test this robot in room 12 of Steele (which will serve as a development laboratory), with 1 person (Pete Trautman), and then increase the number of people and complexity of movement gradually. In this setting we will also test out the remote kill switch on the robot, which enables Pete Trautman, at the touch of a button, to stop the robot.

Once the robot has demonstrated sufficiently safe behavior in room 12 of Steele (collision free trajectories 100% of the time, over 20 hours of testing, with up to 3 people moving in the same environment), we will graduate the robot to Chandler cafeteria.

We will run initial Chandler experiments in the late afternoon when only a few patrons are present, in order to verify the safety of the robot in the new environment when few people are present.

We will begin testing in Chandler during off-peak hours, around 3 p.m. (Chandler cafeteria closes at 3:30 p.m.), when there are only a few people milling around. After we have demonstrated that the robot is sufficiently safe during this time period (again, 100% success rate, for 20 hours of testing), we will graduate again, up to a time period when the average crowd is approximately twice as large. We will continue this "test if robot is sufficiently safe, then graduate" cycle until we reach near peak hours.

Pete Trautman has received the consent of the manager of campus dining operations, Jaime Reyes (reyes@caltech.edu), and the Assistant Vice President of Student Affairs, Housing and Dining, Peter Daily (pdaily@caltech.edu), to mount the cameras.

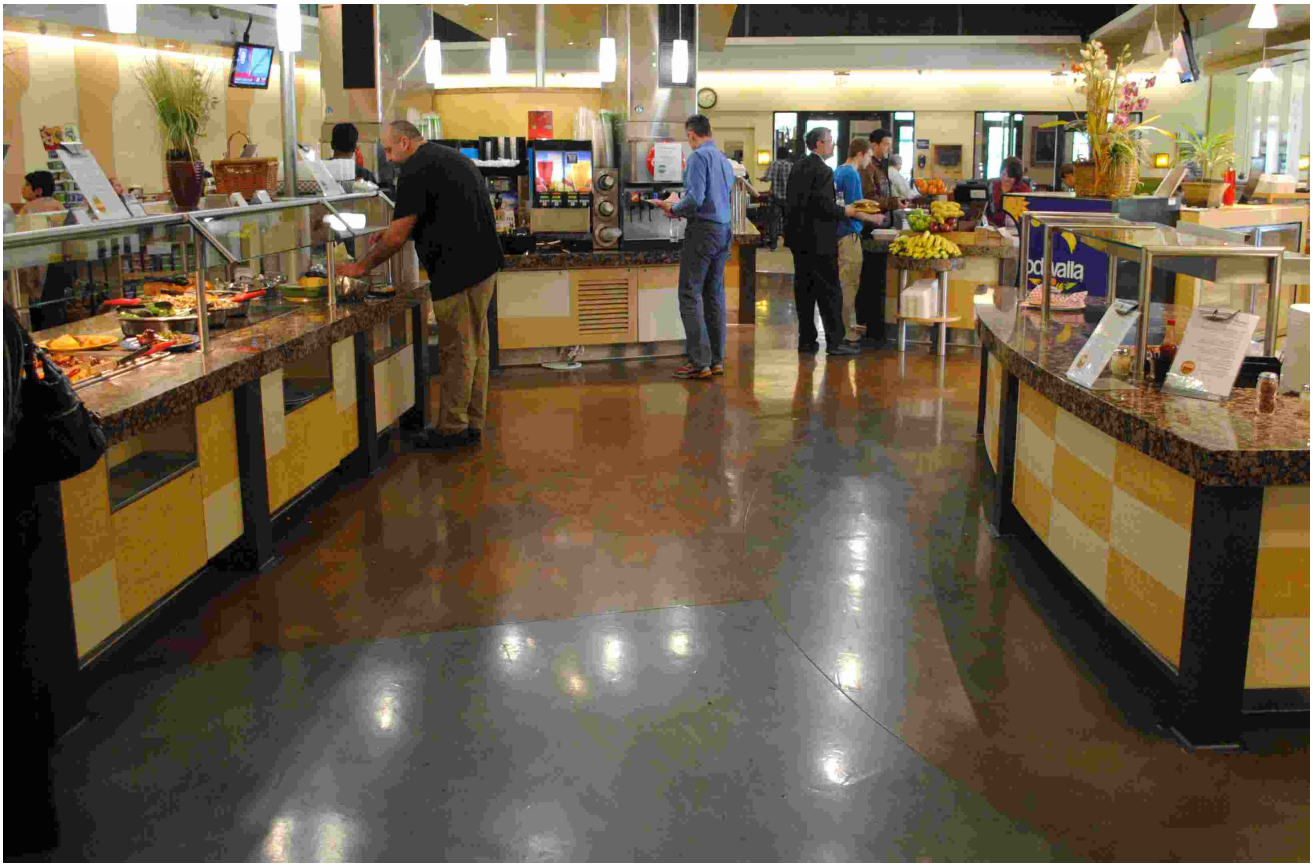


Figure 1: View of Chandler area (taken in front of Italian station) where experiments and filming will occur. The camera will be mounted above this area, and look down on Chandler patrons. See 4 and 3 for diagrammatic information.

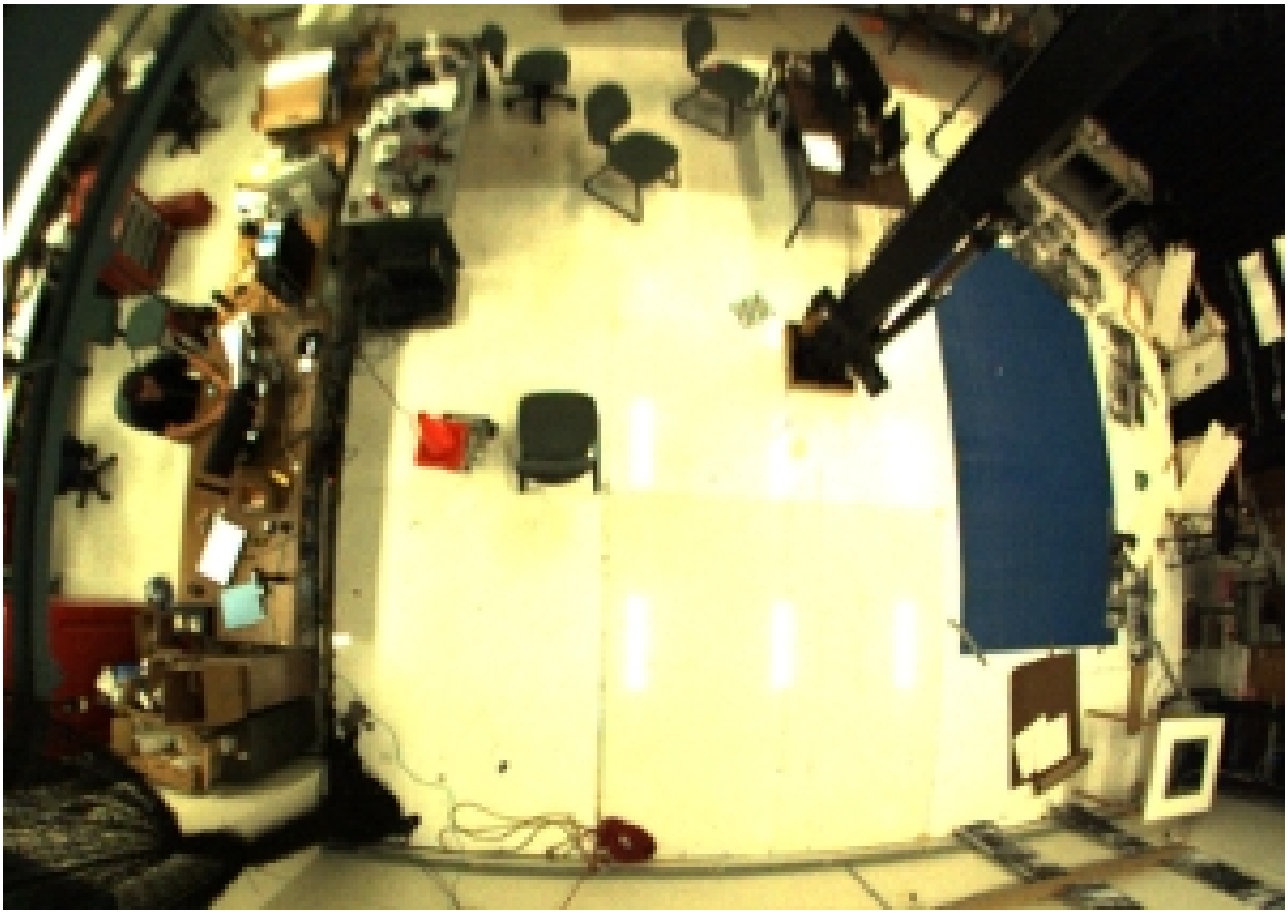


Figure 2: Sample still of human subject from approximately 12 feet. Notice the granularity makes identifying the individual impossible.

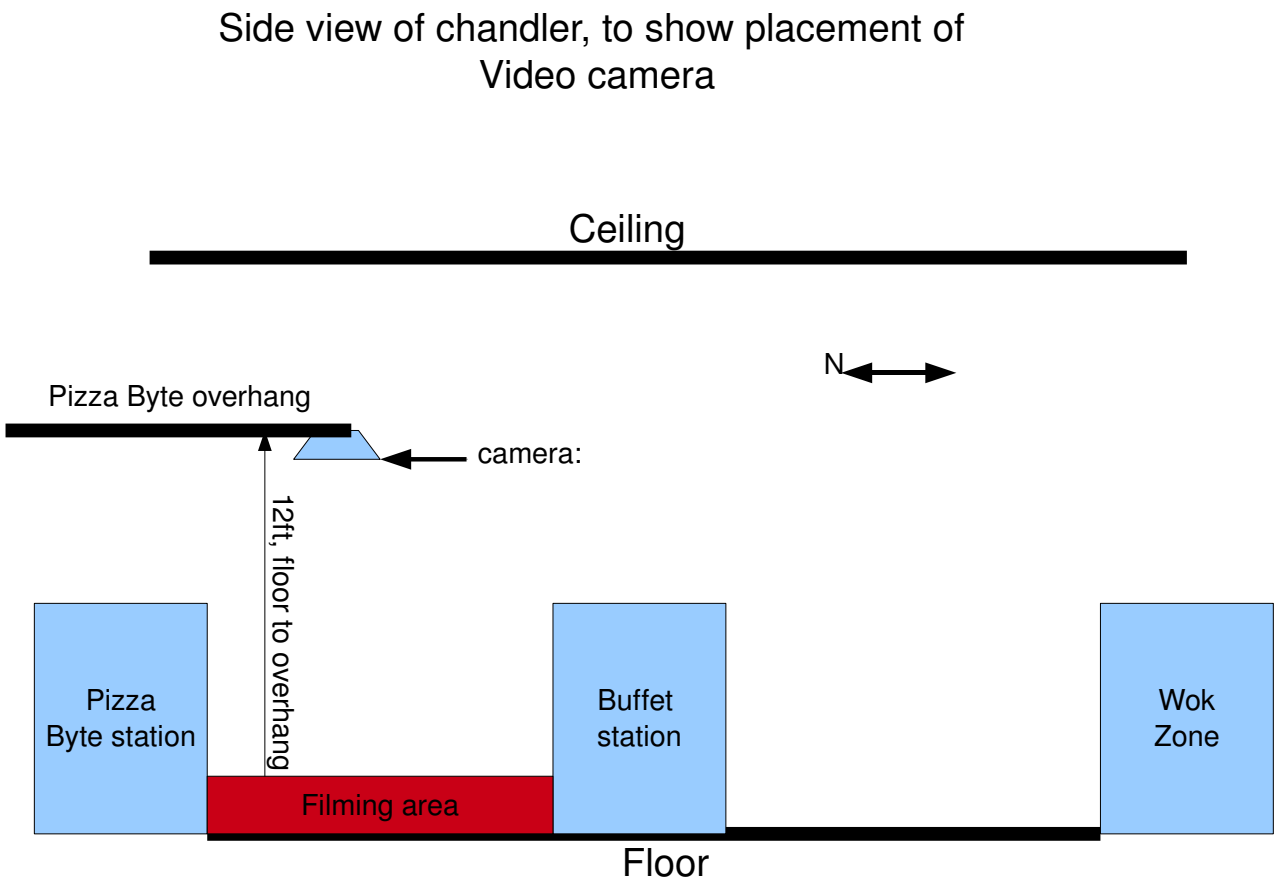


Figure 3: Diagram of filming area in Chandler cafeteria, as viewed from the west side cash register

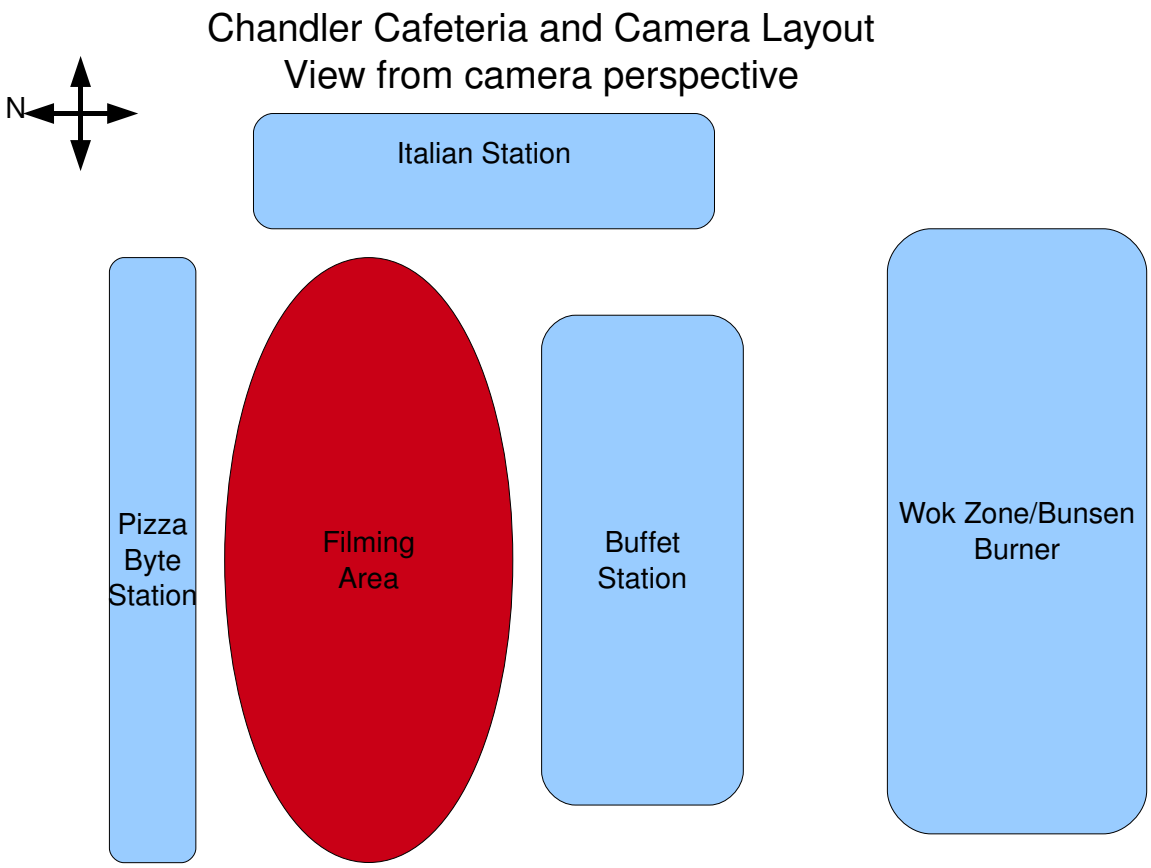


Figure 4: Diagram of filming area in Chandler cafeteria, as viewed from the camera's perspective

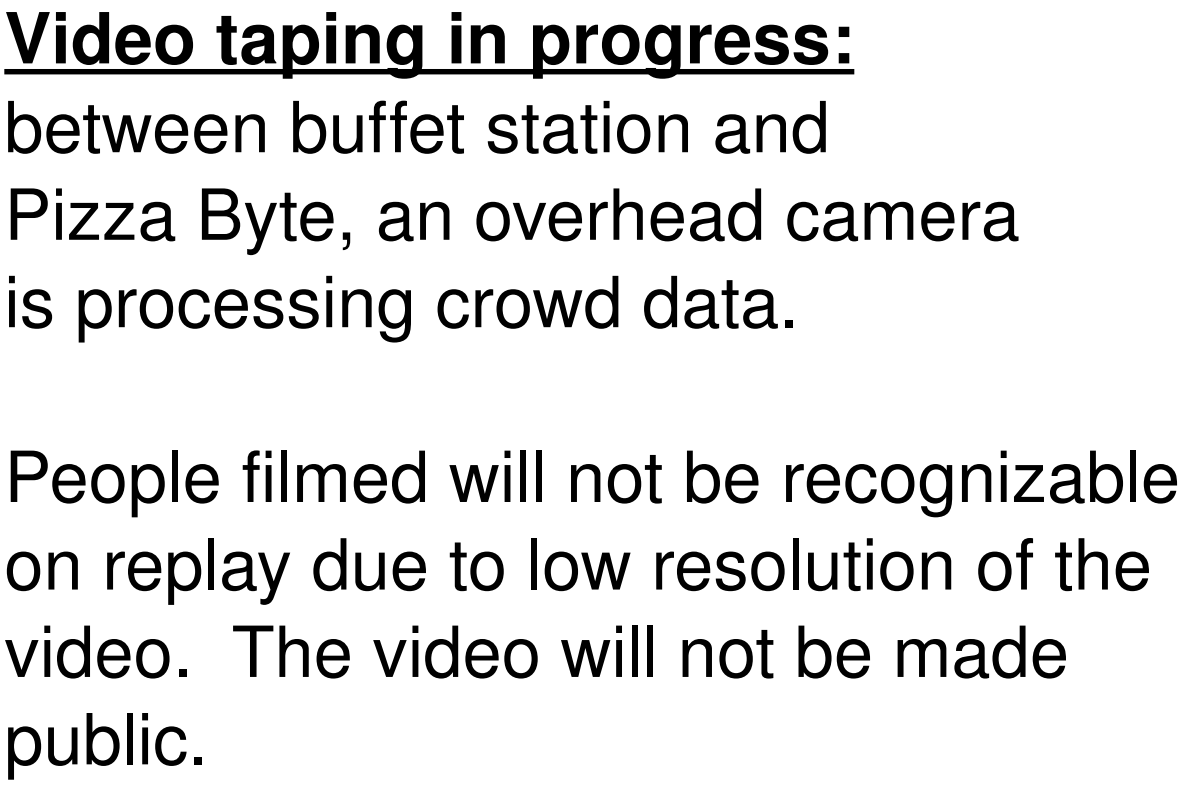


Figure 5: Sign which will be used to inform Chandler patrons of filming; these signs will be placed near the main cash registers, so that anyone entering the filming area will be aware of the filming.

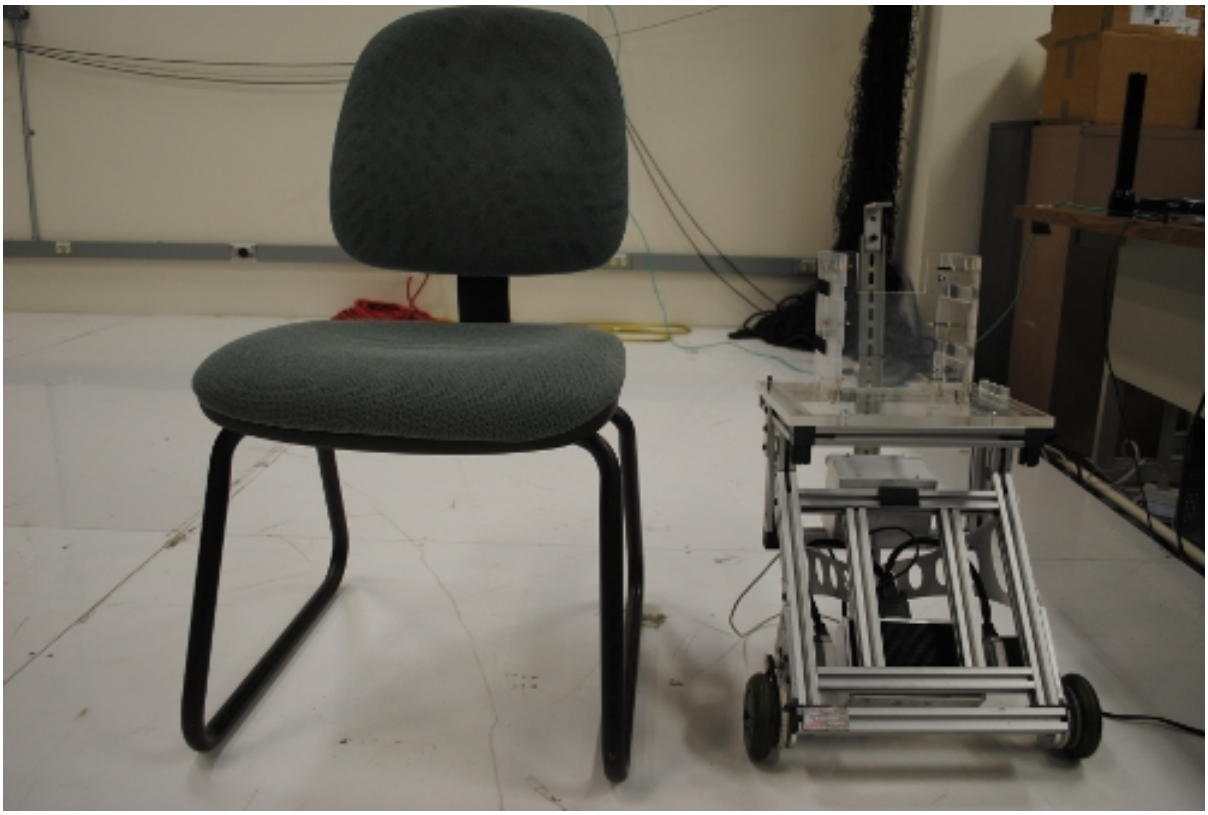


Figure 6: Picture of robot next to a standard office chair.

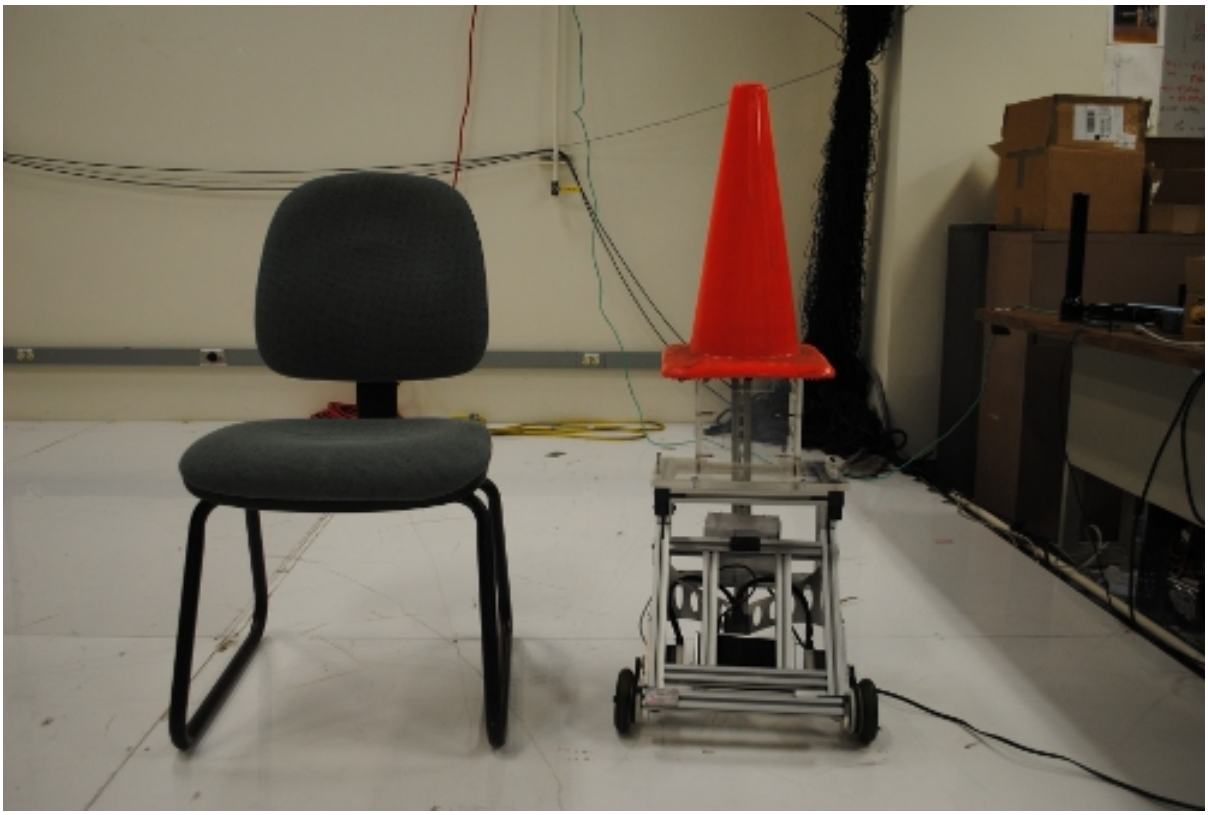


Figure 7: Picture of robot with orange “safety hat” on.

Robot experiment in progress:
between buffet station and
Pizza Byte, a small robot is being
operated (it has an orange cone
on its head).

The robot is under continual
supervision; please treat the robot
as if it were “just another Chandler”
patron.

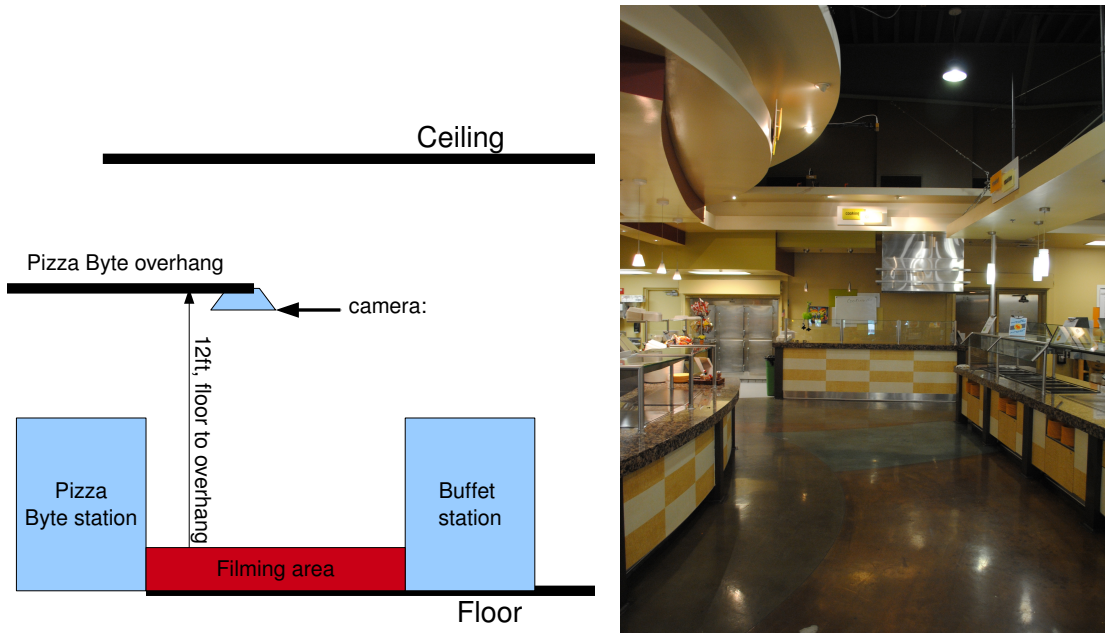
Figure 8: Sign indicating presence of robot in cafeteria; these signs will be placed near the main cash registers, so that anyone entering the experiment area will be aware of the experiment.

Appendix D

Chandler Dining Hall Computational Infrastructure

D.1 Mounting the Cameras

In Figure D.1(a) we present a diagram of how a single camera was mounted in the ceiling of Chandler dining hall, with corresponding reference photograph in Figure D.1(b). In Figures D.2(a), D.2(b), and D.2(c), we see how the cameras were arranged in the ceiling of the cafeteria. The cameras were registered to one another by detecting a special feature (a checkerboard, for instance) in the overlap between pairwise cameras. In particular, the westernmost camera was chosen to be at location $(x, y, z) = (0, 0, 0)$. A large checkerboard was then presented in the overlap of the westermost camera and the middle camera, and triangulation on features was used to determine the coordinate tranformation (both in (x, y, z) and in the three orientation angles). The middle and easternmost cameras were then presented with the large checkerboard pattern in the overlap region, and the triangulation process was repeated. In this way, all three cameras were registered to one another (see Figure D.3 for a pictorial representation of the overhead registration). These transformations were then utilized as described in Section 4.3.1.



(a) (b)

Figure D.1: **(a)** Side diagram of the observation space **(b)** Same perspective as diagram for actual cafeteria

(a) (b) (c)

Figure D.2: **(a)** Image of a single stereo vision camera **(c)** Image of three cameras looking from west to east **(d)** Image of three cameras looking from east to west

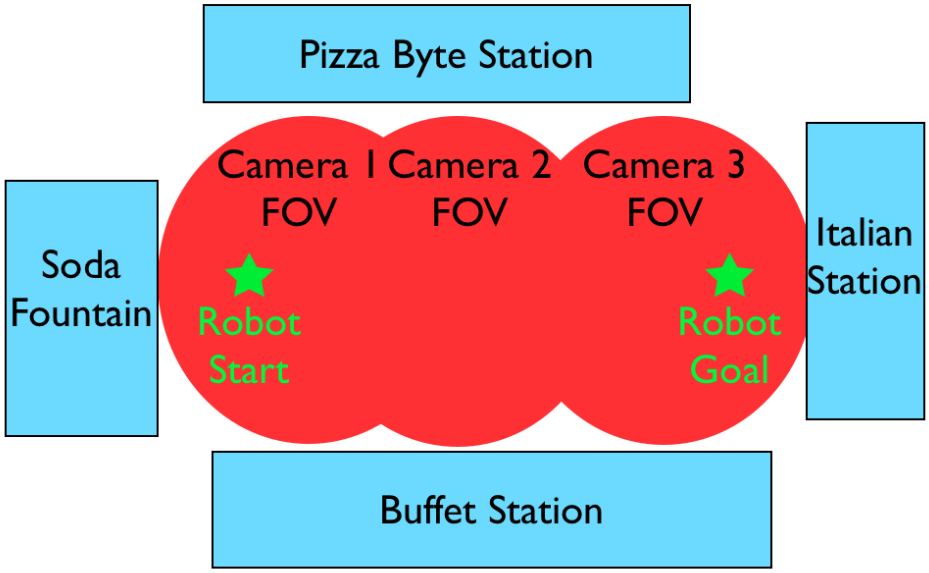


Figure D.3: Overhead diagram of the observation space

D.2 Networking and Powering the Cameras

In Figure D.4 we see an example of how a single camera was networked and powered; in particular, the grey cord coming out of the back of the camera in the image is an IEEE1394b cable, which provides both power and data. The IEEE1394b cable is then run through a firewire to ethernet repeater. From the repeater, 100 meters of ethernet cable is run along the backside of the cafeteria along the overhang (see Figure D.2(b)) to a small cafeteria service closet. The exit point for the ethernet cable is inside of a cafeteria service closet. Inside of the service closet, the ethernet signal was converted back into 1398b data. This data was then fed into a single computer (however, because of the high data rates of each individual Bumblebee2 camera, a dedicated PCI card was required for each camera).

A single Windows machine was responsible for computing the stereo images of all three cameras and extracting the tracks (see Section 4.3.1 for details), as well as writing the data logs to disk. Importantly, running any additional cameras would have required an additional computer, which would have increased the complexity of the system dramatically, since syncing cameras *across* computers is very difficult.

Figure D.4: Camera networking and power. Gray cord is firewire line carrying both.

Bibliography

[1] D. Althoff, Kuffner. J., Wollherr. D., and Buss M. Safety assessment of trajectories for navigation in uncertain and dynamic environmentssment of robot trajectories for navigation in uncertain and dynamic environments. *Autonomous Robots*, 2012.

[2] C. Andrieu, A. Doucet, S.S. Singh, and V.B. Tadic. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 2004.

[3] G. Aoude, B. Luders, and J. How. Sampling-based threat assessment algorithms for intersection collisions involving errant drivers. In *International Federation of Automatic Control Symposium on Intelligent Autonomous Vehicles*, 2010.

[4] G. Aoude, J. Joseph, N. Roy, and J. How. Mobile agent trajectory prediction using bayesian nonparametric reachability trees. In *American Institute of Aeronautics and Astronautics Infotech at Aerospace Conference*, 2011.

[5] G. Aoude, B. Luders, J. Joseph, N. Roy, and J. How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 2011.

[6] R. Arkin, M. Fujita, R. Hasegawa, and T. Takagi. An ethological and emotional basis for human-robot interaction. *Robotics and Autonomous Systems*, 2003.

[7] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 2002.

[8] A. Bauer, K. Klasing, and et al. The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 2009.

[9] A. Bautin, L. Martinez-Gomez, and T. Fraichard. Inevitable collision states: a probabilistic perspective. In *IEEE International Conference on Robotics and Automation*, 2010.

[10] A. Bautin, L. Martinez-Gomez, and T. Fraichard. Inevitable collision states: A probabilistic perspective. In *IEEE International Conference on Robotics and Automation*, 2010.

[11] M. Bennewitz, W. Burgard, G. Cielniak, and S Thrun. Learning Motion Patterns of People for Compliant Robot Motion. *International Journal of Robotics Research*, 2005.

[12] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2005.

[13] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 2005.

[14] C. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, New York, NY, 2006.

[15] L. Blackmore. Robust path planning and feedback design under stochastic uncertainty. In *American Institute of Aeronautics and Astronautics Guidance, Navigation and Control*, 2008.

[16] L. Blackmore and B. Williams. Optimal, robust predictive control of nonlinear systems under probabilistic uncertainty using particles. In *American Control Conference*, 2007.

[17] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference*, 2006.

[18] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation*, 1999.

[19] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Association for the Advancement of Artificial Intelligence*, 1998.

[20] J. Carson. *Robust Model Predictive Control with a Reactive Safety Mode*. PhD thesis, California Institute of Technology, 2008.

[21] A. Castro-Gonzalez, M. Shiomi, T. Kanda, M. Salichs, H. Ishiguro, and N. Hagita. Position prediction in crossing behaviors. In *IEEE International Conference on Intelligent Robots and Systems*, 2010.

[22] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion*. MIT Press, 2005.

[23] W. Dahm. USAF Chief Scientist report on technology horizons: a vision for Air Force science and technology during 2010-2030. Technical report, United States Air Force, 2010.

[24] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation*, 1999.

[25] A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *Oxford Handbook of Nonlinear Filtering*. Oxford Handbooks, 2008. URL `http://www.cs.ubc.ca/~arnaud/doucet_johansen_tutorialPF.pdf`.

[26] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

[27] A. Dragan and S. Srinivasa. Formalizing assistive teleoperation. *Robotics: Science and Systems*, 2012.

[28] N. Du Toit. *Robotic Motion Planning in Dynamic, Cluttered, Uncertain Environments: the Partially Closed-loop Receding Horizon Control Approach*. PhD thesis, Caltech, 2009.

[29] N. Du Toit and J. Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 2011.

[30] N. Du Toit and J. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 2012.

[31] E. Marder Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *IEEE International Conference on Robotics and Automation*, 2010.

[32] S. Fiore, N. Badler, L. Boloni, M. Goodrich, A. Wu, and J. Chen. Human-robot teams collaborating socially, organizationally, and culturally. *2011 Human Factors and Ergonomics Society Symposium – From Teleoperation to Teammate: Applying Theory and Method from the Cognitive and Computational Sciences to Create Human-Robot Teams*, 2011.

[33] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 1998.

[34] A. Foka and P. Trahanias. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *International Journal of Social Robotics*, 2010.

[35] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 1997.

[36] T. Fraichard. A short paper about motion safety. In *IEEE International Conference on Robotics and Automation*, 2007.

[37] T. Fraichard and H. Asama. Inevitable collision states: a step towards safer robots? In *IEEE International Conference on Intelligent Robots and Systems*, 2003.

[38] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *IEEE International Conference on Robotics and Automation*, 2007.

[39] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *IEEE International Conference on Intelligent Robots and Systems*, 2008.

[40] C. Fulgenzi, Anne Spalanzani, and Christian Laugier. Probabilistic motion planning among moving obstacles following typical motion patterns. In *IEEE International Conference on Intelligent Robots and Systems*, 2009.

[41] D. Glas, S. Satake, T. Kanda, and N. Hagita. An interaction design framework for social robots. In *Robotics: Science and Systems*, 2011.

[42] E. Hall. A system for notation of proxemic behavior. *American Anthropologist*, 1963.

[43] E. Hall. *The Hidden Dimension*. Doubleday, 1966.

[44] B. Hardin and M. Goodrich. On using mixed-initiative control: A perspective for managing large-scale robotic teams. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2009.

[45] K. Hayashi, M. Shiomi, T. Kanda, and N. Hagita. Friendly patrolling: A model of natural encounters. In *Robotics: Science and Systems*, 2011.

[46] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 1995.

[47] D. Helbing, I. Farkas, and T. Viscek. Simulating dynamical features of escape panic. *Nature*, 2000.

[48] D. Helbing, P. Molnar, I. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Environment and Planning B: Planning and Design*, 2001.

[49] H. Helble and S. Cameron. 3-D path planning and target trajectory prediction for the oxford aerial tracking system. In *IEEE International Conference on Robotics and Automation*, 2007.

[50] P. Henry, C. Vollmer, B. Ferris, and D. Fox. Learning to navigate through crowded environments. In *IEEE International Conference on Robotics and Automation*, 2010.

[51] S.M. Herman. *A Particle Filtering Approach to Joint Passive Radar Tracking and Target Classification*. PhD thesis, University of Illinois, 2002.

[52] J. Joseph, F. Doshi-Velez, and N. Roy. A bayesian non-parametric approach to modeling mobility patterns. *Autonomous Robots*, 2011.

[53] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1960.

[54] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro. Interactive robots as social partners and peer tutors. In *International Conference on Human-Computer Interaction*, 2004.

[55] T. Kanda, D. Glas, M. Shiomi, H. Ishiguro, and N. Hagita. Who will be the customer?: a social robot that anticipates people's behavior from their trajectories. In *International Conference on Ubiquitous Computing*, 2008.

[56] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, 1990.

[57] J. Ko and D. Fox. GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. In *IEEE International Conference on Intelligent Robots and Systems*, 2008.

[58] J. Ko and D. Fox. Learning GP-BayesFilters via Gaussian process latent variable models. In *Robotics: Science and Systems*, 2009.

[59] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[60] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, 1991.

[61] T. Kruse and et al. Exploiting human cooperation in human-centered robot navigation. In *IEEE International Symposium on Robots and Human Interactive Communications*, 2010.

[62] T. Kruse, A. Kirsch, E. Sisbot, and R. Alami. Dynamic generation and execution of human aware navigation plans. In *International Conference on Autonmous Agents and Multiagent Systems*, 2010.

[63] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems*, 2012.

[64] C. Lam, C. Chou, K. Chiang, and L. Fu. Human centered robot navigation: Towards a harmonious human-robot coexisting environment. *IEEE Transactions on Robotics*, 2011.

[65] F. Large, D. Vasquez, T. Fraichard, and C. Laugier. Avoiding cars and pedestrians using velocity obstacles and motion predictionotion prediction. In *Intelligent Vehicles Symposium*, 2004.

[66] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[67] S. LaValle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 2001.

[68] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[69] H. Li, Yi Wu, and H. Lu. Visual tracking using particle filters with gaussian process regression. In *Proceedings of the Pacific Rim Symposium on Advances in Image and Video Technology*, 2009.

[70] M. Luber, G. Tipaldi, and K. Arras. People tracking with human motion predictions from social forces. In *IEEE International Conference on Robotics and Automation*, 2010.

[71] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 2000.

[72] R. Mead and M.J. Matarić. A probabilistic framework for autonomous proxemic control in situated and mobile human-robot interaction. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2012.

[73] R. Mead, A. Atrash, and M.J. Matarić. Proxemic feature recognition for interactive robots: automating metrics from the social sciences. In *International Conference on Social Robotics*, 2011.

[74] R. Mead, A. Atrash, and M.J. Matarić. Recognition of spatial dynamics for predicting social interaction. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2011.

[75] T. P. Minka. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, 2001.

[76] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *Association for the Advancement of Artificial Intelligence*, 2002.

[77] M. Morari and J. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, 1999.

[78] R. Murphy. Findings from NSF-JST-NIST workshop on rescue robotics. In *IEEE International Workshop on Safety Security and Rescue Robotics*, 2010.

[79] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: modeling social behavior for multi-target tracking. In *International Conference on Computer Vision*, 2009.

[80] S. Pellegrini, A. Ess, M. Tanaskovic, and L. Van Gool. Wrong turn - no dead end: a stochastic pedestrian motion model. In *International Workshop on Socially Intelligent Surveillance and Monitoring*, 2010.

[81] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 2003.

[82] A. Powers and S. Kiesler. Tracing people's mental model from a robot's physical attributes. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2006.

[83] N. Pradhan, T. Burg, and S. Birchfield. Robot crowd navigation using predictive position fields in the potential function framework. In *American Control Conference*, 2011.

[84] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL `http://www.gaussianprocess.org/gpml/`.

[85] K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: Science and Systems*, 2012.

[86] B. Reeves and C. Nass. *The media equation: how people treat computers, television, and new media like real people and places*. Cambridge University Press, 1996.

[87] J. Rios-Martinez, A. Spalanzani, and C. Laugier. Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach. In *IEEE International Conference on Intelligent Robots and Systems*, 2011.

[88] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, Boston, MA, 2004.

[89] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *IEEE International Conference on Robotics and Automation*, 1999.

[90] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. *IEEE Intelligent Systems*, 1999.

[91] N. Roy, G. Gordon, and S. Thrun. Planning under uncertainty for reliable health care robotics. In *Field and Service Robotics*, 2003.

[92] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2005.

[93] L. Saiki, S. Satake, R. Huq, D. Glas, T. Kanda, and N. Hagita. How do people walk side-by-side? Using a computational model of human behavior for a social robot. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2012.

[94] J. Schulte, C. Rosenberg, and S. Thrun. Spontaneous, short-term interaction with mobile robots. In *IEEE International Conference on Robotics and Automation*, 1999.

[95] D. Feil Seifer and M.J. Matarić. Human-robot interaction. In *Encyclopedia of Complexity and Systems Science*. Springer, 2009.

[96] D. Feil Seifer and M.J. Matarić. People-aware navigation for goal-oriented behavior involving a human partner. In *International Conference on Development and Learning*, 2011.

[97] D. Feil Seifer, K. Skinner, and M.J. Matarić. Benchmarks for evaluating socially assistive robotics. *Interaction Studies: Psychological Benchmarks of Human-Robot Interaction*, 2007.

[98] M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita. Interactive humanoid robots for a science museum. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2006.

[99] M. Shiomi, T. Kanda, D. Glas, S. Satake, H. Ishiguro, and N. Hagita. Field trial of networked social robots in a shopping mall. In *IEEE International Conference on Intelligent Robots and Systems*, 2009.

[100] C. Sidner and C. Lee. Engagement rules for human-robot collaborative interactions. *IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[101] R. Siegwart, K. Arras, S. Bouabdhalla, D. Burnier, G. Froidevaux, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis. Robox at expo.02: A large scale installation of personal robots. *Robotics and Autonomous Systems*, 2003.

[102] J. Snape, J. van den Berg, S. Guy, and D. Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 2011.

[103] S. Srinivasa, D. Ferguson, J. Vandeweghe, R. Diankov, D. Berenson, C. Helfrich, and K. Strasdat. The robotic busboy: Steps towards developing a mobile robotic home assistant. In *IEEE International Conference on Intelligent Robots and Systems*, 2008.

[104] M. Svenstrup, T. Bak, and J. Andersen. Trajectory planning for robots in dynamic human environments. In *IEEE International Conference on Intelligent Robots and Systems*, 2010.

[105] L. Takayama and C. Pantofaru. Influences on proxemic behaviors in human-robot interaction. In *IEEE International Conference on Intelligent Robots and Systems*, 2009.

[106] L. Takayama, D. Dooley, and W. Ju. Expressing thought: Improving robot readability with animation principles. In *ACM/IEEE International Conference on Human-Robot Interaction*, 2011.

[107] Y. Teh. Dirichlet processes. In *Enclyclopedia of Machine Learning*. Springer, 2010.

[108] S. Thompson, T. Horiuchi, and S. Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *International Conference on Autonomous Robots and Agents*, 2009.

[109] S. Thrun, M. Beetz, and et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 2000.

[110] S. Thrun, J. Schulte, and C. Rosenberg. Interaction with mobile robots in public places. *IEEE Intelligent Systems*, 2000.

[111] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[112] M. Toussaint. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning*, 2009.

[113] P. Trautman. Probabilistic tools for human robot cooperation. In *ACM/IEEE International Conference on Human-Robot Interaction: Human Agent Robot Teamwork Workshop*, 2012.

[114] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense interacting crowds. In *IEEE International Conference on Intelligent Robots and Systems*, 2010.

[115] J. van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation*, 2008.

[116] J. van den Berg, S. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *International Symposium on Robotics Research*, 2009.

[117] J. van den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal of Robotics Research*, 2011.

[118] Kevin Waugh, Brian D. Ziebart, and J. Andrew (Drew) Bagnell. Computational rationalization: The inverse equilibrium problem. In *International Conference on Machine Learning*, 2010.

[119] B. D. Ziebart, A. Maas, A. Dey, and J. A. Bagnell. Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior. In *International Conference on Ubiquitous Computing*, 2008.

[120] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *IEEE International Conference on Intelligent Robots and Systems*, 2009.