Real-Time Predictive Modeling and Robust Avoidance of Pedestrians with Uncertain, Changing Intentions

by

Sarah Kathryn Ferguson

S.B., Aerospace Engineering
Massachusetts Institute of Technology (2012)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Masters of Science in Aerospace Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

Signature redacted

Author .....
Department of Aeronautics and Astronautics
May 22, 2014

Signature redacted

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Accepted by .....
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Real-Time Predictive Modeling and Robust Avoidance of Pedestrians with Uncertain, Changing Intentions

by

Sarah Kathryn Ferguson

Submitted to the Department of Aeronautics and Astronautics
on May 22, 2014, in partial fulfillment of the
requirements for the degree of
Masters of Science in Aerospace Engineering

Abstract

To plan safe trajectories in urban environments, autonomous vehicles must be able to interact safely and intelligently with other dynamic agents. Due to the inherent structure of these environments, drivers and pedestrians tend to exhibit a common set of motion patterns. The challenges are therefore to learn these motion patterns such that they can be used to predict future trajectories, and to plan safe paths that incorporate these predictions.

This thesis considers the modeling and robust avoidance of pedestrians in real time. Pedestrians are particularly difficult to model, as their motion patterns are often uncertain and/or unknown *a priori*. The modeling approach incorporates uncertainty in both intent (i.e., where is the pedestrian going?) and trajectory associated with each intent (i.e., how will he/she get to this location?), both of which are necessary for robust collision avoidance. A novel changepoint detection and clustering algorithm (Changepoint-DPGP) is presented to enable quick detection of changes in pedestrian behavior and online learning of new behaviors not previously observed in prior training data. The resulting long-term movement predictions demonstrate improved accuracy in terms of both intent and trajectory prediction, relative to existing methods which consider only intent or trajectory.

An additional contribution of this thesis is the integration of these predictions with a chance-constrained motion planner, such that trajectories which are probabilistically safe to pedestrian motions can be identified in real-time. Hardware components and relevant control and data acquisition algorithms for an autonomous test vehicle are implemented and developed. Experiments demonstrate that an autonomous mobile robot utilizing this framework can accurately predict pedestrian motion patterns from onboard sensor/perception data and safely navigate within a dynamic environment.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank a number of people for contributing to this thesis and my career at MIT. First and foremost, I would like to thank my advisor, Professor Jonathan How, for his guidance and support in developing me as a researcher. Jon taught me to approach research in a methodical way, by focusing on high-level problem definition and the identification of key challenges, such that solutions to meaningful problems can be developed. I am grateful to have had an advisor so strongly committed to my success and growth as a student.

I would also like to thank all of the members of the Aerospace Controls Lab, whose support and feedback has been integral to my research. I am especially indebted to Brandon Luders, with whom I have worked so closely on this project. Brandon has been constantly available to answer questions, provide feedback, and inspire new ideas. Without his help, guidance, and support, both personally and professionally, this work would not have been possible. I am also grateful to Mark Cutler for his hardware and ROS expertise, and incredible patience as I learned to integrate these components. An additional thanks to everyone else who participated in the crowd-sourced editing of this thesis, especially Trevor Campbell, Jack Quindlen, Rob Grande, Steven Chen, Aaron Ellertson, Chris Lowe, and Ali Agha.

Finally, I would like to thank my wonderful and supportive family and friends, whose love and encouragement truly made this thesis possible. To my parents and sisters, thank you for your unconditional support and for always encouraging me to reach for my dreams. To my friends from MIT, especially Aaron Koski and Reguli Granger, you have become like a second family to me. Thank you for being there through the highs and lows. Getting through the past few years would not have been possible (or nearly as fun) without you.

This research has been funded by Ford Motor Company, through the Ford-MIT Alliance. The author also acknowledges Boeing Research & Technology for support of the test environment in which experiments were conducted.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous vehicles operating in complex urban environments must be able to interact safely and intelligently with human drivers and pedestrians. A major challenge in planning safe trajectories through these environments is the limited ability to accurately anticipate the future trajectories of these and other dynamic agents, as they move according to a variety of complex factors (e.g., internal state, desires) that are not directly observable. Due to the inherent structure of urban environments, drivers and pedestrians tend to exhibit a common set of mobility patterns, which are directly observable via state estimates.

The challenges are therefore to learn these motion patterns such that they can be used to predict future trajectories, and to plan safe paths that avoid future collisions by incorporating these predictions. While existing probabilistic planning frameworks can readily admit dynamic agents with uncertain future trajectory distributions [6], these agents typically demonstrate complex motion patterns that make modeling future motion and quantifying uncertainty difficult.

Consider the pedestrian crosswalk scenario depicted in Figure 1-1. In order to reliably navigate the crosswalk, the autonomous vehicle must predict both the underlying intent of the pedestrian (i.e., continuing along the sidewalk vs. traversing the crosswalk, as indicated in green) and the possible trajectories corresponding to each

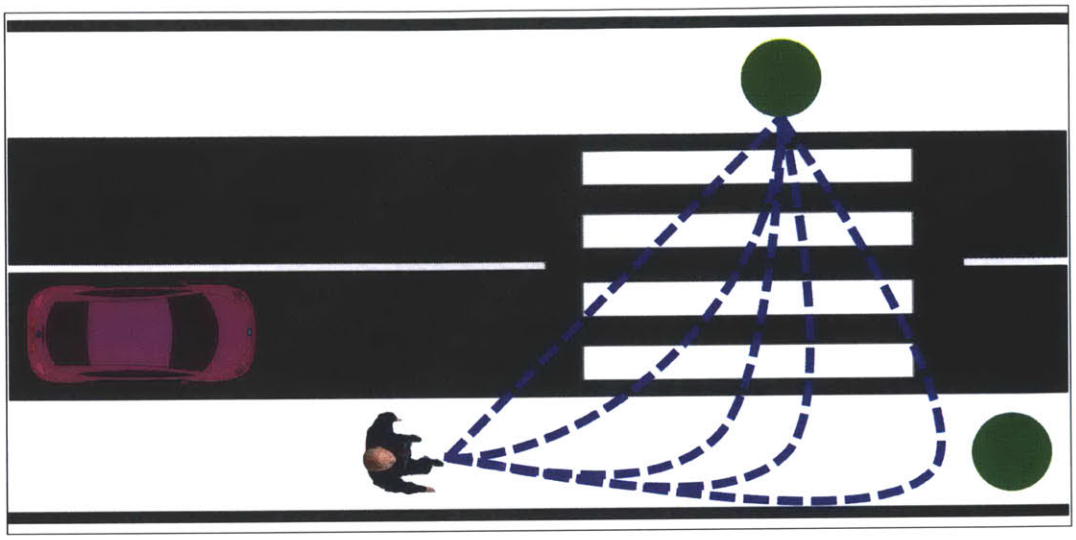


Figure 1-1: Pedestrian crosswalk scenario, demonstrating uncertainty in intent (green) and path (blue)

intent. For example, even if it is somehow known that the pedestrian will cross the street, his specific future trajectory is still unknown (i.e., it could be one of any of the paths to the goal indicated in blue). Even with perfect sensing, long-term prediction algorithms must incorporate both forms of uncertainty to enable safe planning [6, 9].

Pedestrians present particular technical challenges in the generation of long-term predictions due to their agility and relatively unrestrictive dynamic and inertial constraints. Specifically, pedestrians demonstrate (i) many unique behaviors, which may not have been previously observed, and (ii) instantaneous changes in motion behavior following changes in intent. The primary objective of this thesis is the development of a modeling framework that accurately predicts the future behavior of agile agents, such that an autonomous vehicle can identify safe trajectories that avoid collision at current and future time steps. Such a framework must be able to learn new behaviors online and update predictions in the presence of changes in intent, while converging to the correct intent prediction as more observations are gathered, capabilities not currently present in existing algorithms.

1.2 Related Work

Existing approaches for modeling and predicting the future trajectories of pedestrians and other dynamic agents can be classified into two categories: those that focus solely on the pedestrian (Section 1.2.1) and those that model an interaction between the pedestrian and autonomous vehicle (Section 1.2.2).

1.2.1 Pedestrian Modeling

Pedestrian modeling techniques can be further classified into dynamics- and pattern-based approaches.

Dynamics-based Approaches

Dynamics-based approaches predict future trajectories by propagating an agent's dynamics forward in time from the current state, typically using a continuous Bayes filter such as the Kalman filter [49]. This approach is particularly popular in target tracking literature. Examples include the Interacting Multiple Modal Kalman filter (IMM-KF), which selects a propagation from a bank of continuously-updated Kalman filters by matching the agent's current mode of operation [35], and the Bayesian Occupancy filter, which applies a Bayes filter to a four-dimensional occupancy grid representation of the agent's state space [14]. Though useful for short-term predictions, the performance of these approaches degrades with increasing time horizons because environmental features (e.g., obstacles, goal points) are not considered.

Pattern-based Approaches

Rather than directly modeling internal features, pattern-based approaches assume that pedestrians tend to follow typical motion patterns, which can be learned and used to predict future trajectories. Long-term predictions are therefore conditioned both on state observations and a library of learned behavior models, rather than on state observations alone. In practice, knowledge of motion patterns within the

environment is seldom available *a priori*, so a variety of machine learning techniques are applied to extract motion patterns from available training data.

The most common approaches are based on the Markov property. This set of approaches include hidden Markov models, in which the hidden state is pedestrian intent [11, 29, 55]; growing hidden Markov models to allow for online learning [52]; and partially observable Markov decision processes to choose actions based on a distribution over pedestrian intents [8]. Because the future state prediction depends only on the current state, these approaches are quick to react to changes in intent. However, for relatively infrequent changes in intent, the Markov assumption can be overly restrictive, as it prevents these algorithms from becoming more certain of pedestrian intent with additional observations.

Gaussian process (GP) approaches have been demonstrated to be well-suited for modeling pedestrian motion patterns, as they perform well with noisy observations and have closed-form predictive uncertainty [18, 19, 28, 45]. Additionally, recent work using GP mixture models enables predictions that account for both intent and trajectory uncertainty [6]. Both sets of approaches use the entire observed trajectory in the prediction of future state, such that certainty in demonstrated intent tends to converge over time. Therefore, when changes in intent occur, these approaches are much slower to detect a change than Markov-based approaches. Additionally, existing GP classification approaches are too slow for online learning of previously unobserved behavior patterns.

The weakness of most of these approaches is that uncertainty in intent is not typically considered; instead, the maximum likelihood trajectory prediction is used for motion planning [8]. Bandyopadhyay et al. [8] model a distribution over possible pedestrian intents using a variant of the Partially Observable Markov Decision Process (POMDP), but use a simple model for trajectory prediction that assumes pedestrians approximately follow the shortest path to their goals. Aoude et al. [6] consider uncertainty in both intent and trajectory, with a GP model for trajectory prediction; however, predictions are slow to recognize changes in intent and online learning of new behaviors is not possible.

1.2.2 Pedestrian/Vehicle Modeling

The previous section reviews pedestrian modeling approaches that are agnostic of the vehicle state; this section considers approaches in which an explicit interaction between the pedestrian and vehicle is modeled. These approaches can be classified into pursuit evasion and cooperative approaches.

Pursuit-Evasion Approaches

Pursuit-evasion approaches represent a worst-case class of predictions, in which the dynamic agent is assumed to be actively trying to collide with the autonomous vehicle [32, 37]. The predicted future trajectory of the dynamic agent is therefore the solution to a differential game, in which the dynamic agent is a pursuer and the autonomous vehicle is an evader [5]. These approaches do provide a lower bound on safety [30], but lead to inherently conservative and unrealistic solutions in an urban setting.

Cooperative Approaches

Cooperative approaches are those in which an explicit degree of cooperation between the dynamic agent and autonomous vehicle is modeled, motivated by navigation in crowded environments with high pedestrian density. Several cooperative approaches utilize inverse reinforcement learning (IRL). Ziebart et al. [56] pedestrian trajectories are obtained from a database via IRL, then the autonomous vehicle navigates such that the predicted pedestrian trajectory is minimally disrupted; Henry et al. [23] IRL is applied to generate human-like behaviors for the autonomous vehicle enabling seamless integration of the vehicle into crowded human environments; Waugh et al. [54] a cooperation model is learned from human trajectory data. Trautman et al. [50] use GPs to model the interaction between dynamic agents and the autonomous vehicle, encoding the notion of mutual avoidance such that the autonomous vehicle will not become stuck.

1.3 Outline and Summary of Contributions

This thesis proposes a novel changepoint detection and clustering algorithm (Changepoint-DPGP) which retains the trajectory prediction accuracy of existing GP approaches while expanding their capabilities. Coupled with offline unsupervised learning of a Gaussian process mixture model (DPGP) [28], this approach enables quick detection of changes in intent and online learning of motion patterns not seen in prior training data. The resulting long-term movement predictions demonstrate improved accuracy relative to offline learning alone in both intent and trajectory prediction.

These predictions can also be used within a chance-constrained motion planner [34] to identify probabilistically safe trajectories in real-time. In experimental results, the proposed algorithm is used to predict the motion of multiple dynamic agents detected from a variety of onboard and external sensors, enabling an autonomous rover to robustly navigate dynamic environments.

- Chapter 2 presents preliminaries for pedestrian modeling including Gaussian processes (GPs), the GP motion model, and the algorithm for batch learning of motion patterns (DPGP) [28].

- Chapter 3 considers application of a chance-constrained motion planning algorithm (CC-RRT) to several motion planning domains of interest. This chapter extends existing work, in which motion patterns are manually defined by an expert [6], by clustering training trajectories into representative motion patterns using DPGP.

- Chapter 4 proposes the Changepoint-DPGP (CP-DPGP) algorithm for online changepoint detection and clustering of observed trajectories given a preliminary set of models learned via DPGP. This algorithm leverages the efficient hypothesis testing framework for changepoint detection and clustering developed in [21] to enable quick classification and online learning of observed behaviors. Simulation results demonstrate improved intent and trajectory prediction accuracy.

- Chapter 5 details the hardware test platform used in real-time experiments, including the test environment with motion capture cameras for object detection and projection technology for display of planning and predictions in the real world. A skid-steer rover with sensing capabilities for obstacle detection and autonomous navigation is presented, in addition to a pure-pursuit control algorithm for trajectory following with modifications for skid-steer dynamics. Two methods for obstacle detection from 2D and 3D lidar are also proposed and evaluated.

- Chapter 6 contributes the real-time demonstration of the proposed prediction and probabilistic motion planning algorithms on hardware in dynamic environments. By embedding predictions of dynamic obstacles in a CC-RRT planner, an autonomous rover is able to safely navigate around pedestrians, robots, and other dynamic obstacles, even in the presence of changing intents and new behaviors. Obstacles are detected both from motion capture cameras and from onboard sensors, demonstrating the ability of this approach to be used within perception-driven planning.

Chapter 2

Preliminaries

This chapter presents preliminaries for Gaussian processes, modeling of pedestrian motion patterns, and batch learning of motion patterns.

2.1 Gaussian Processes

This section summarizes relevant definitions and properties related to Gaussian processes, obtained from Rasmussen and Williams [45] unless otherwise noted. A Gaussian process (GP) is defined as a collection of random variables, any finite subset of which has a joint Gaussian distribution with mean $\hat{\mu}(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$. Correlation between points is defined by the squared exponential covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\top L(\mathbf{x} - \mathbf{x}')\right) + \sigma_n^2 \delta(\mathbf{x}, \mathbf{x}'), \tag{2.1}$$

where $\delta(\mathbf{x}, \mathbf{x}') = 1$ if $\mathbf{x} = \mathbf{x}'$ and zero otherwise. This choice enforces a high correlation between the outputs and nearby inputs. Specifically, the matrix $L = \text{diag}(\mathbf{l})^{-2}$ is a diagonal matrix of positive length-scale parameters l_i which informally represent the distance between points along each axis of the input space required for function values to become uncorrelated.

The σ_n term represents within-point variation (e.g. due to noisy measurements); the ratio of σ_n and σ_f denotes the relative effects of noise and influences from nearby

points. Together with L, these terms represent the set of hyperparameters θ^{GP}. Note that the covariance function models the correlation between data points, whereas the hyperparameters are interpreted from the data; therefore, each GP has a unique set of hyperparamters that can be learned (e.g. [53]).

As is common in the GP literature, it is assumed that the GP has a zero mean. In general, this assumption is not limiting since the posterior estimate of the latent function is not restricted to zero. The elements of the GP kernel matrix $K(X, X)$ are defined as $K_{i,j} = k(x_i, x_j)$, and $k(X, x_{i+1}) \in \mathbb{R}^i$ denotes the kernel vector corresponding to the $i + 1^{th}$ measurement. The joint distribution is given by

$$\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(Z,Z) + \omega_n^2 I & k(Z, z_{i+1}) \\ k^T(Z, z_{i+1}) & k(z_{i+1}, z_{i+1}) \end{bmatrix}\right). \tag{2.2}$$

The conditional probability can then be calculated as a normal variable with mean

$$\hat{\mu}(x_{i+1}) = \alpha^T k(X, x_{i+1}), \tag{2.3}$$

where $\alpha = [K(X,X) + \omega_n^2 I]^{-1} y$ are the kernel weights, and covariance

$$\Sigma(x_{i+1}) = k(x_{i+1}, x_{i+1}) - k^T(X, x_{i+1})[K(X,X) + \omega_n^2 I]^{-1} k(X, x_{i+1}). \tag{2.4}$$

Due to the addition of the positive definite matrix $\omega_n^2 I$, the matrix inversion in (2.3) and (2.4) is well defined.

2.2 Motion Patterns and Modeling

This section details the motion model applied to pedestrian motion, as previously presented in [6, 28].

2.2.1 Gaussian Process Motion Patterns

A trajectory is represented as a set of observed locations in two-dimensional space (x_1^i, y_1^i), (x_n^i, y_n^i), ..., $(x_{L^i}^i, y_{L^i}^i)$, where L^i is the total length of the trajectory t^i of agent i. Because it is assumed that trajectories are collected from sensor data, there are no restrictions on trajectory length or discretization (i.e. trajectories need not be of the same length, and the time steps between each observation may be irregular).

A motion pattern is defined as a mapping from each location (x^i, y^i) to a distribution over trajectory derivatives $\left(\frac{\Delta x^i}{\Delta t}, \frac{\Delta y^i}{\Delta t}\right)$, resulting in a velocity flow-field in $x - y$ space. Because the predicted next position of an agent can be obtained from its current position and the trajectory derivative at that location, modeling trajectories is equivalent to modeling trajectory derivatives. This representation enables the grouping of trajectories with similar velocity field characteristics into representative motion patterns, regardless of the trajectory length or discretization.

GPs can be used to effectively model motion patterns. The GP serves as a non-parametric form of interpolation between the discrete trajectory measurements comprising each motion pattern. Specifically, given an observed (x, y) location, the GP predicts the trajectory derivatives at that location. Because the space is two-dimensional, each motion pattern is modeled by a pair of GPs, each mapping (x, y) location to trajectory derivatives in the x or y direction. Figure 2-1 presents trajectory derivates (blue) sampled at points on a discrete grid for a GP defined by six pedestrian trajectories (black), demonstrating how the velocity flow field allows for generalization of predictions to any initial state.

Referring to the discussion of GPs in Section 2.1, the assumption of zero mean encodes the prior bias that agents are expected to remain stationary in the absence of additional knowledge (in the form of training trajectories) at a particular location; the squared exponential covariance function ensures that similar trajectories will result in similar predictions [28].

GPs generalize well to regions of sparse data while avoiding the problem of overfitting in regions of dense data. Additionally, GP models are robust to unaligned,

noisy measurements and are well-suited for modeling the continuous paths underlying potentially non-uniform time-series samples of pedestrian locations [45]. Although standard GPs have significant computational cost, efficient methods for online use have been developed (e.g. [15, 6]).

2.2.2 Motion Model

The motion model is defined as a finite mixture of GP motion patterns weighted by their probability. The finite mixture model defines a distribution over the ith observed trajectory t^i

$$p(t^i) = \sum_{j=1}^{M} p(b_j)p(t^i|b_j), \tag{2.5}$$

where b_j is the jth motion pattern, $p(b_j)$ is its prior probability of the jth motion pattern, and $p(t^i|b_j)$ is the probability of trajectory t^i given b_j. The number of motion patterns M can be learned offline via an automated clustering process [28], and in this thesis, is incremented as new behavior patterns are identified online.

The posterior probability of b_j given a target trajectory t^i is described by

$$p(b_j|t^i) \propto p(t^i|b_j)p(b_j), \tag{2.6}$$

where the distribution $p(t^i|b_j)$ is

$$\begin{aligned} p(t^i|b_j) &= \prod_{t=0}^{L_i} p\left(\frac{\Delta x_t}{\Delta t}\middle| x^i_{0:t}, y^i_{0:t}, \{t^k : z = j\}, \theta^{GP}_{x,j}\right) \cdot \\ &= p\left(\frac{\Delta y_t}{\Delta t}\middle| x^i_{0:t}, y^i_{0:t}, \{t^k : z = j\}, \theta^{GP}_{y,y}\right), \end{aligned} \tag{2.7}$$

where z_k indicates the motion pattern to which trajectory t^k is assigned, $\{t^k : z = j\}$ is the set of all trajectories assigned to motion pattern j, and $\theta^{GP}_{x,t}, \theta^{GP}_{y,t}$ are the hyperpameters of the GP defining motion pattern b_j.

The prior $p(b_j)$ is initialized to be proportional to the number of trajectories in

motion pattern j, relative to the total number of trajectories in all motion patterns. It is updated after each prediction with the previous posterior probability.

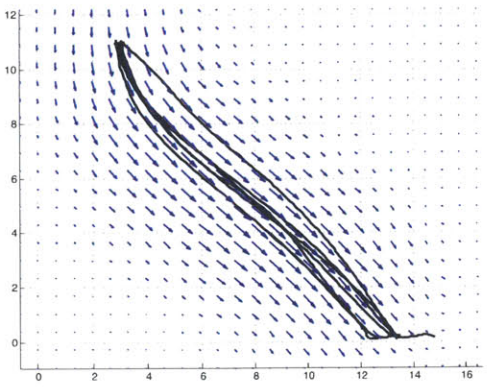


Figure 2-1: Trajectory derivates for GP

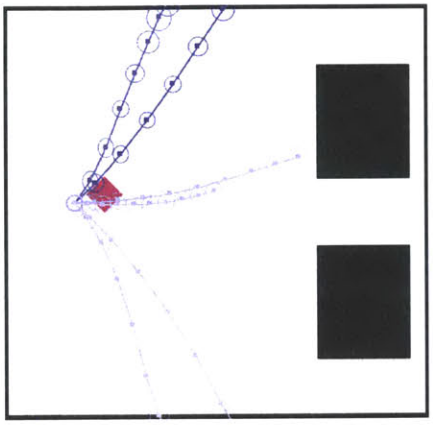
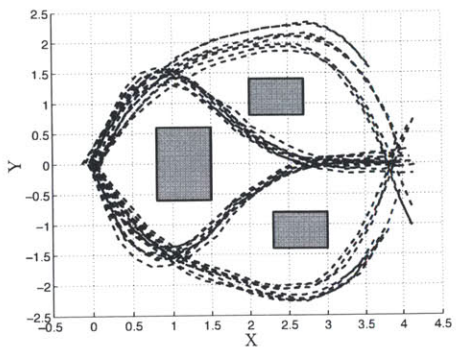


Figure 2-2: Trajectory predictions from the finite motion model

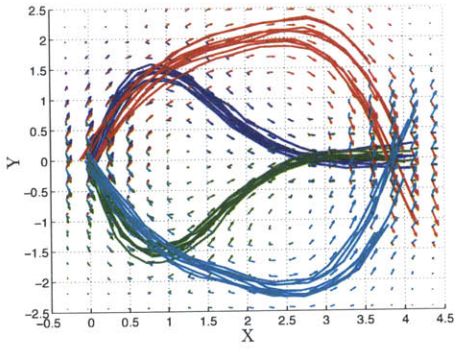
2.2.3 Estimation of Future Trajectories

Future pedestrian trajectories are predicted for each motion pattern using the approach of [16, 20]. This approach provides a fast, analytic approximation of the GP output given a distribution over the input. In this application, the distribution over the input corresponds to the uncertain position of the agent; specifically, the uncertainty distribution from a prediction at time $t-1$ is incorporated at time t to estimate the agent's position at $t+1$. Representative predictions are presented in Figure 2-2, where points are mean predicted position, ellipses are propagated uncertainty, and a darker shade of blue indicates higher likelihood.

This approach allows for an analytic prediction of the future position K time steps into the future that is more efficient that the sampling strategy used by Aoude et al. [6], reducing the number of queries to the GP from $N \cdot K$ to K where N is the number of trajectories in the GP motion pattern [28].



(a) Unlabeled trajectories



(b) Clustered trajectories with corresponding GP flow fields, where color indicates cluster assignment

Figure 2-3: DPGP clustering result given initial set of unlabeled trajectories

2.3 Batch Learning of Motion Patterns

It is expected that observed pedestrian trajectories will demonstrate a variety of qualitatively different behaviors. These behavior motion patterns are learned from an input set of unlabeled trajectories by DPGP, a Bayesian nonparametric clustering algorithm that automatically determines the most likely number of clusters without *a priori* information [28]. This section reviews the DPGP algorithm, which is used to cluster observed pedestrian trajectories into representative motion patterns in batch.

The DPGP algorithm models motion patterns as Gaussian processes, as described in Section 2.2.1. The motion model is a mixture of motion patterns weighted by Dirichlet process (DP) mixture weights. The DP is a distribution of discrete distributions in which the number of motion patterns is potentially unbounded. The concentration parameter α controls the probability of new cluster formation, such that smaller values of α result in fewer clusters. Intuitively, the parameter α enforces the notion that there are a few motion patterns that agents tend to exhibit; therefore, trajectories are more likely to fit existing clusters than to form new ones. In practice, α can be considered as an additional hyperparameter to be learned from the data.

The prior probability that trajectory t^i has an assignment z_i to an existing motion

pattern b_j is

$$p(z_i = j|z_{-i}, \alpha) = \frac{n_j}{N - 1 + \alpha}, \tag{2.8}$$

where z_{-i} refers to the motion pattern assignments for the remaining trajectories, n_j is the number of trajectories currently assigned to b_j, and N is the total number of trajectories. The probability that trajectory t^i will be assigned to a new motion pattern is

$$p(z_i = M + 1||z_{-i}, \alpha) = \frac{\alpha}{N - 1 + \alpha}, \tag{2.9}$$

where M is the total number of motion patterns.

The probability of cluster assignment for trajectory t^i is obtained from the DP prior (Equations 2.8 and 2.9) and probability of motion pattern b_j given t^i (Equation 2.7). Specifically, the probability that trajectory t^i will be assigned to an existing motion pattern is

$$p(z_i = j|t^i, \alpha, \theta^{GP}_{x,j}, \theta^{GP}_{y,j}) \propto p(t^i|b_j) \left(\frac{n_j}{N - 1 + \alpha} \right), \tag{2.10}$$

and the probability that trajectory t^i will be assigned to a new motion pattern is

$$p(z_i = M + 1|t^i, \alpha) \propto \int p(t^i|b_j) d\theta^{GP}_{x,j} d\theta^{GP}_{y,j} \left(\frac{\alpha}{N - 1 + \alpha} \right), \tag{2.11}$$

Because exact inference over the space of GPs and DPs is intractable, samples are drawn from this posterior distribution using Gibbs sampling techniques. At each iteration, the DP hyperparameter α is resampled and the GP hyperparameters for the j behavior patterns $\theta^{GP}_{x,j}, \theta^{GP}_{y,j}$ are set to their maximum likelihood values given the current trajectory clustering. For each trajectory, the assignment z_i is drawn from Equations 2.10 and 2.11.

Figure 2-3 demonstrates a representative clustering produced by the DPGP algorithm given unlabeled trajectories. Full details of the DPGP algorithm are presented

in [28].

Chapter 3

Motion Planning with DPGP

This chapter considers the integration of the chance-constrained rapidly exploring random tree (CC-RRT) motion planning algorithm with DPGP. This is a direct extension of the work in [6], in which an expert manually clusters trajectories into representative mobility patterns. By incorporating DPGP, this motion planning framework becomes much more accessible, as mobility patterns can be learned from the observed data rather than manually defined.

First, the CC-RRT algorithm developed in [34] is presented (Section 3.1). As shown, CC-RRT readily admits the inclusion of dynamic agents while maintaining probabilistic feasibility by representing future trajectories as a time-parameterized Gaussian mixture model. A sampling-based algorithm for reachability refinement of GPs (RR-GP) developed in [6] is presented in Section 3.2. This algorithm allows for the refinement of sparse GP predictions, improving predictive accuracy. DPGP is combined with CC-RRT and RRGP (Section 3.3) to form a motion planning framework that is applied to the problem of navigation through structured environments to enable an autonomous vehicle to safely avoid moving threats (Section 3.4).

3.1 Problem Statement

This section presents the motion planning problem for dynamic obstacles with uncertain future positions; further details are found in [6]. Consider the noisy LTI

discrete-time dynamics

$$x_{t+1} = f(x_t, u_t, w_t), \tag{3.1}$$

$$w_t \sim \mathcal{N}(0, P_w), \tag{3.2}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state vector, $u_t \in \mathbb{R}^{n_u}$ is the input vector, and $w_t \in \mathbb{R}^{n_w}$ is an i.i.d. process noise uncertainty acting on the system, unknown at current and future time steps but with the known unbounded probability distribution (3.2). Here $\mathcal{N}(\widehat{a}, P_a)$ represents a Gaussian random variable with mean $\widehat{a}$ and covariance P_a.

The system is additionally subject to the state and input constraints

$$x_t \in \mathcal{X}_t \equiv \mathcal{X} \backslash \mathcal{X}_{1t} \backslash \cdots \backslash \mathcal{X}_{n_o t}, \tag{3.3}$$

$$u_t \in \mathcal{U}, \tag{3.4}$$

where $\mathcal{X}, \mathcal{X}_{1t}, \ldots, \mathcal{X}_{n_o t} \subset \mathbb{R}^{n_x}$ are convex polytopes, $\mathcal{U} \subset \mathbb{R}^{n_u}$, and the $\backslash$ operator denotes relative complement (set difference). The sets $\mathcal{X}$ and $\mathcal{U}$ define a set of time-invariant convex constraints acting on the state and input, respectively. The sets $\mathcal{X}_{1t}, \ldots, \mathcal{X}_{n_o t}$ represent n_o convex, polytopic obstacles to be avoided. The time dependence of $\mathcal{X}_t$ in (3.3) allows the inclusion of both static and dynamic obstacles, allowing the representation of dynamic pedestrians in this work. For each obstacle, the shape and orientation are assumed to be known, while the placement is uncertain. This is represented as

$$\mathcal{X}_{jt} = \mathcal{X}_j^0 + c_{jt}, \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \tag{3.5}$$

$$c_{jt} \sim \mathcal{N}(\widehat{c}_{jt}, P_{c_{jt}}), \quad \forall\, j \in \mathbb{Z}_{1,n_o}, \tag{3.6}$$

where the $+$ operator denotes set translation and $\mathbb{Z}_{a,b}$ represents the set of integers between a and b inclusive. In this model, for the jth obstacle, $\mathcal{X}_j^0 \subset \mathbb{R}^{n_x}$ is a convex polytope of known, fixed shape and orientation, while $c_{jt} \in \mathbb{R}^{n_x}$ represents an uncertain and/or time-varying translation.

A dynamic agent with uncertain future position is assumed to follow one of M possible behaviors, as determined by the DPGP algorithm. As described in Section 2.2.2, the position distribution incorporates uncertainty in future intent and trajectory given intent, with future trajectories computed as described in Section 2.2.3. At each timestep, the probability of collision with dynamic agent j can therefore be expressed as a weighted sum of the collision probabilities under each behavior. The construction maintains all existing probabilistic guarantees by treating the trajectory distribution associated with each behavior as a separate time-parameterized obstacle with the resulting risk scaled by the intent likelihood of that behavior [6].

The primary objective of the motion planning problem is to reach some goal region $\mathcal{X}_{\text{goal}} \subset \mathbb{R}^{n_x}$ while ensuring the input constraints (3.4) are satisfied, while the state constraints (3.3) are probabilistically satisfied. This is represented via path-wise and time-step-wise chance constraints; respectively,

$$\mathbb{P}\left(\bigwedge_t x_t \in \mathcal{X}_t\right) \geq \delta_p, \qquad \mathbb{P}\left(x_t \in \mathcal{X}_t\right) \geq \delta_s, \quad \forall\, t, \tag{3.7}$$

where $\mathbb{P}(\cdot)$ denotes probability, $\bigwedge$ represents a conjunction over the indexed constraints, and $\delta_s, \delta_p \in [0.5, 1]$.

3.2 RR-GP

The RR-GP algorithm learns motion pattern models by combining Gaussian process (GP) predictions with a sampling-based reachability refinement, which conditions the GP predictions to enforce dynamic and environmental constraints [6]. By doing so, the accuracy of the behavior and trajectory predictions is significantly increased without having to increase the GP resolution. This algorithm can therefore be used to refine sparse GP predictions, resulting in decreased computational load as compared to dense GP predictions of comparable accuracy.

Figure 3-1 provides a visual illustration of the RR-GP approach from [6]. Gaussian processes are learned for each behavior from observations of agents demonstrating

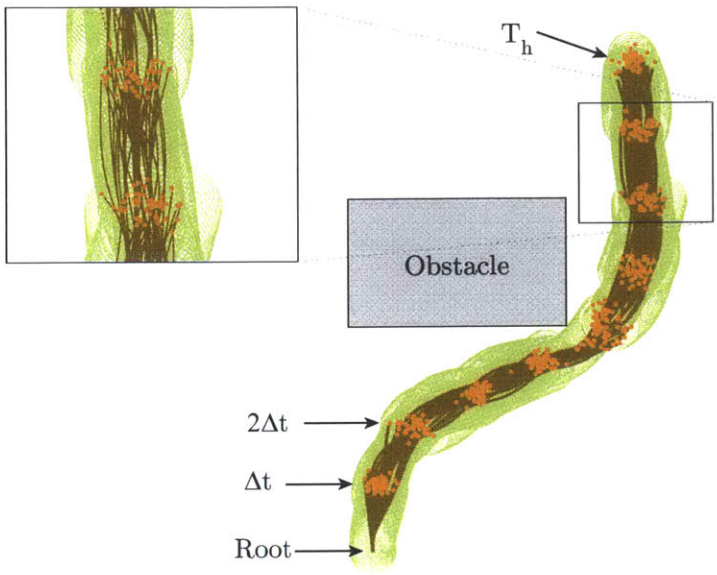


Figure 3-1: Illustration of RR-GP algorithm (Source: [6])

that behavior, which are used as labeled training trajectories. Samples from those GPs (orange dots) are taken at fixed-timestep intervals for each motion. A tree of trajectories (brown) is then generated using those samples, taking into account the actual size of the dynamic obstacle (green circle) and environmental obstacles it is expected to avoid (gray). Since all trajectories remaining in the tree must be dynamically feasible and satisfy all environmental constraints, the remaining samples provide a conditioned estimate of the dynamic obstacle predictions at each timestep.

3.3 Integration of DPGP

The integration of the DPGP algorithm into this framework is straightforward. Given a set of training trajectories, the DPGP algorithm clusters the trajectories into M behavior patterns and learns a GP representation for each behavior pattern. The number of behavior patterns M does not need to be pre-specified, and is instead learned from the data itself. The addition of DPGP thus automates the process of determining the motion model for each environment, improving the applicability of the framework in [6] to new environments.

After the motion model has been determined, the algorithm steps are as follows. Sparse (1 sec) predictive position distributions are generated for the learned motion model as described in Section 2.2.3. Recall that these position distributions represent uncertainty in both intent and path. The trajectory distribution for each intent is refined by the RR-GP algorithm to impose dynamic and environmental constraints. The resulting behavior distribution is modeled as a time-parameterized obstacle with risk scaled by the intent prediction of that behavior, enabling CC-RRT to generate safe trajectories to avoid the current and future predicted positions of the dynamic agent, as described previously (Section 3.1).

3.4 Application: Navigation in Structured Environments

Several examples are now provided demonstrating the ability of the CC-RRT planner to safely avoid a dynamic obstacle via DPGP predictions. In these examples, the DPGP predictions are enhanced with the reachability-based refinement of RR-GP [6].

The autonomous vehicle is modeled as a double integrator,

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ v^x_{t+1} \\ v^y_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ v^x_t \\ v^y_t \end{bmatrix} + \begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u^x_t + w^x_t \\ u^y_t + w^y_t \end{bmatrix},$$

where $dt = 0.1$s, subject to avoidance constraints $\mathcal{X}$ (including velocity bounds) and input constraints

$$\mathcal{U} = \{(u^x, u^y) \mid |u^x| \leq 4, |u^y| \leq 4\}.$$

To emphasize the impact of the dynamic obstacle's uncertainty, the host vehicle's own dynamics are assumed deterministic: $w^x_t \equiv w^y_t \equiv 0$. Trajectories are simulated

and executed in closed-loop via the controller

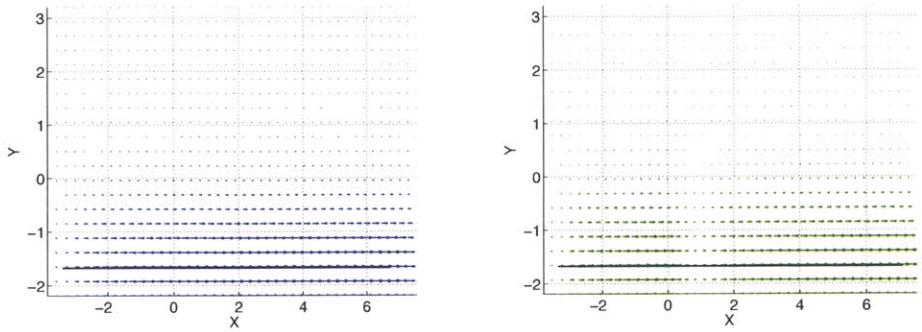
$$\begin{aligned} u_t^x &= -1.5(x_t - r_t^x) - 3(v_t^x - r_t^{v_x}), \\ u_t^y &= -1.5(y_t - r_t^y) - 3(v_t^y - r_t^{v_y}), \end{aligned}$$

where (r_t^x, r_t^y) is the reference position and $(r_t^{v_x}, r_t^{v_y})$ is the reference velocity; the reference r_t is moved continuously between waypoints at a fixed speed of 0.35 m/s. The speed of the target vehicle is capped at 0.4 m/s.

The dynamic agent trajectories are pre-generated for each behavior by having a human operator manually drive a simulated vehicle through the environment via a wireless joystick, tuned to emulate traditional, nonlinear control of an automobile. Ten trajectories are collected for each behavior pattern at 50 Hz. Five trajectories for each behavior are used by DPGP to train the motion model (Figure 3-2); the remaining trajectories are randomly sampled to select the dynamic agent trajectory. There are four possible behaviors for the target vehicle as soon as it reaches the intersection: (a) left turn, (b) left turn after stopping for 1 s, (c) straight, and (d) straight after stopping for 1 s.

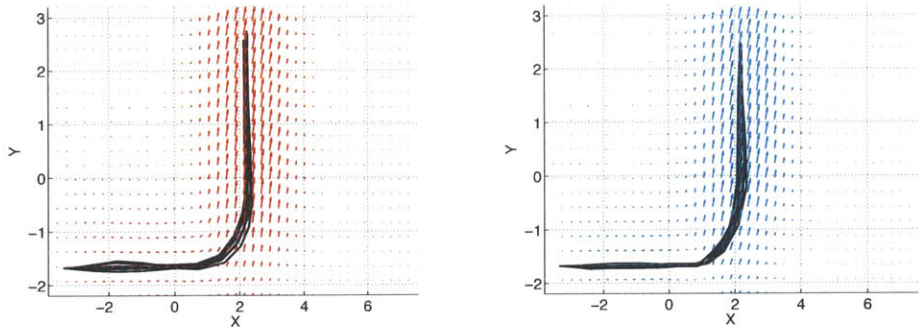
The first scenario is the intersection scenario presented in Figure 3-3. The autonomous vehicle's path history and current path are in orange, while the vehicle itself is represented by a large orange circle. The tree and planned path are for the most part denoted in green; however, they are shaded by risk such that riskier nodes/branches are more red. The autonomous vehicle's objective is to reach the goal position (green circle) while avoiding all static obstacles (black) and the dynamic target vehicle (magenta diamond). Note that the lane boundaries are represented as static obstacles to enforce realistic constraints on the vehicle. The blue paths indicate the paths predicted by the RR-GP algorithm for each possible behavior, including 2σ uncertainty ellipses; more likely paths are indicated with a brighter shade of blue.

Initially, the planner gives the autonomous vehicle a path to have it cross the intersection (Figure 3-3a). However, as the target vehicle approaches the intersection (Figure 3-3b) and its possible behaviors become distinct, the planner curtails the host



(a) Straight

(b) Straight stop



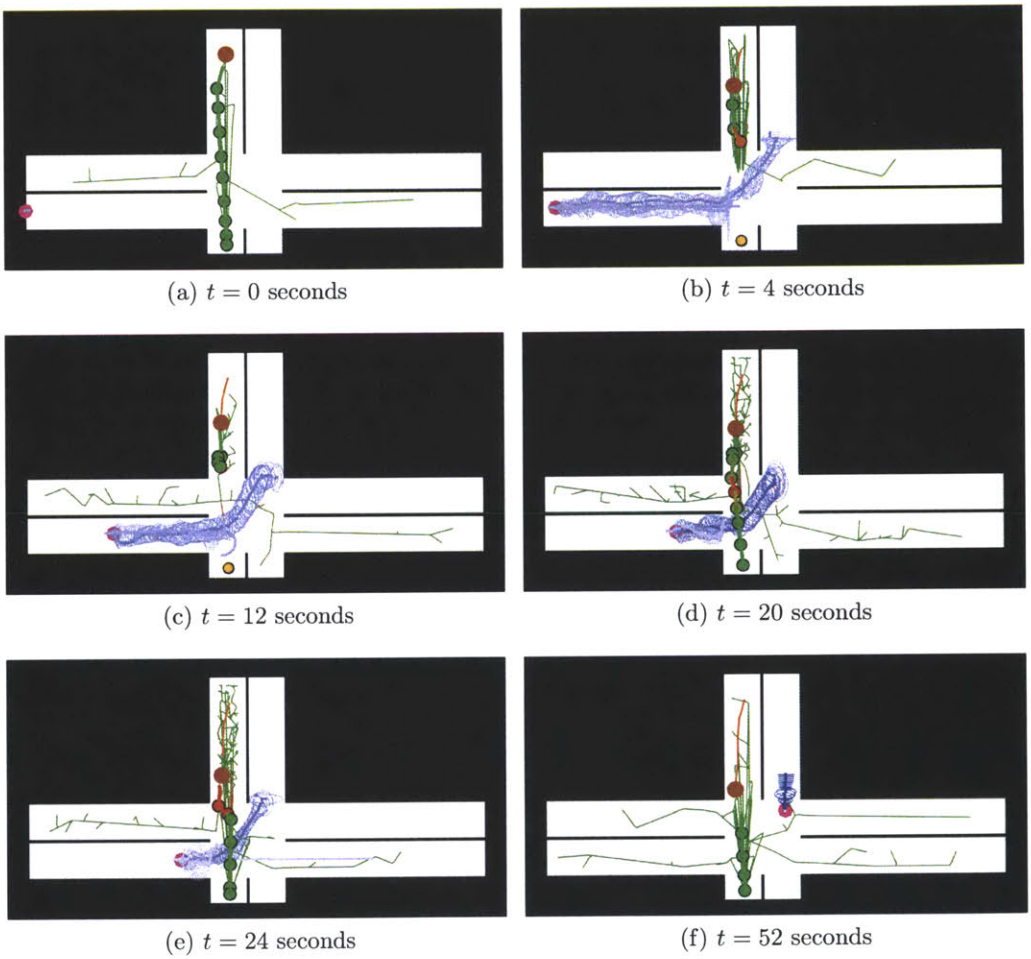
(c) Turn

(d) Turn stop

Figure 3-2: Intersection behavior models resulting from DPGP clustering

vehicle's path to stop at the intersection entry. The final node in the path places the autonomous vehicle too close to both the lane boundary constraint and predictive distribution, so it is removed (Figure 3-3c). As the intent likelihood of the target vehicle converges, the autonomous vehicle plans a path through the intersection, accounting for the risk introduced by the uncertainty in future position of the target vehicle (Figures 3-3d and 3-3e). Once it is clear that the target vehicle is turning left, the planner notes that the intersection crossing has relatively little risk (Figure 3-3f) and the autonomous vehicle follows a straight trajectory to eventually reach the goal.

Figure 3-4 and 3-5 give two examples for the obstacle field scenario, in which the host vehicle is moving left-to-right while avoiding a dynamic obstacle moving in the opposite direction. The dynamic obstacle has six possible behaviors, corresponding to which of the three corridors it traverses in the central passage, and the two possible



(a) $t = 0$ seconds

(b) $t = 4$ seconds

(c) $t = 12$ seconds

(d) $t = 20$ seconds

(e) $t = 24$ seconds

(f) $t = 52$ seconds

Figure 3-3: Representative snapshots of integrated DPGP and CC-RRT algorithms, modified intersection scenario

to which of the three corridors it traverses in the central passage, and the two possible operating speeds of the agent (slow and fast).

In Figure 3-4, the CC-RRT planner quickly finds a path all the way to the goal before the obstacle has begun to move (Figure 3-4a). However, as the dynamic obstacle begins moving, the DPGP predictions quickly detect that the obstacle is planning to traverse the bottom corridor, curtailing the host vehicle's planned safe path (Figure 3-4b). However, the planner quickly finds an alternate route through the central corridor (Figure 3-4c); though this path overlaps with several possible behaviors, the low likelihood of those behaviors (indicated by the very light blue shading in Figure 3-4c results in a risk of collision below the necessary threshhold. As the host vehicle begins executing this path and the DPGP predictions become more confident, the path is refined to reach the goal more quickly (Figure 3-4d). In this case, the planner path for the host vehicle overlaps with the target vehicle's most likely path, and comes close to the target vehicle's current location. However, because the prediction model anticipates that the target vehicle will have continued moving left to the central corridor by the time the host vehicle arrives, the path is known to be safe. Indeed, the host vehicle is able to continue executing this path to reach the goal safely (Figures 3-4e and 3-4f). The second example proceeds in a similar manner (Figure 3-5).

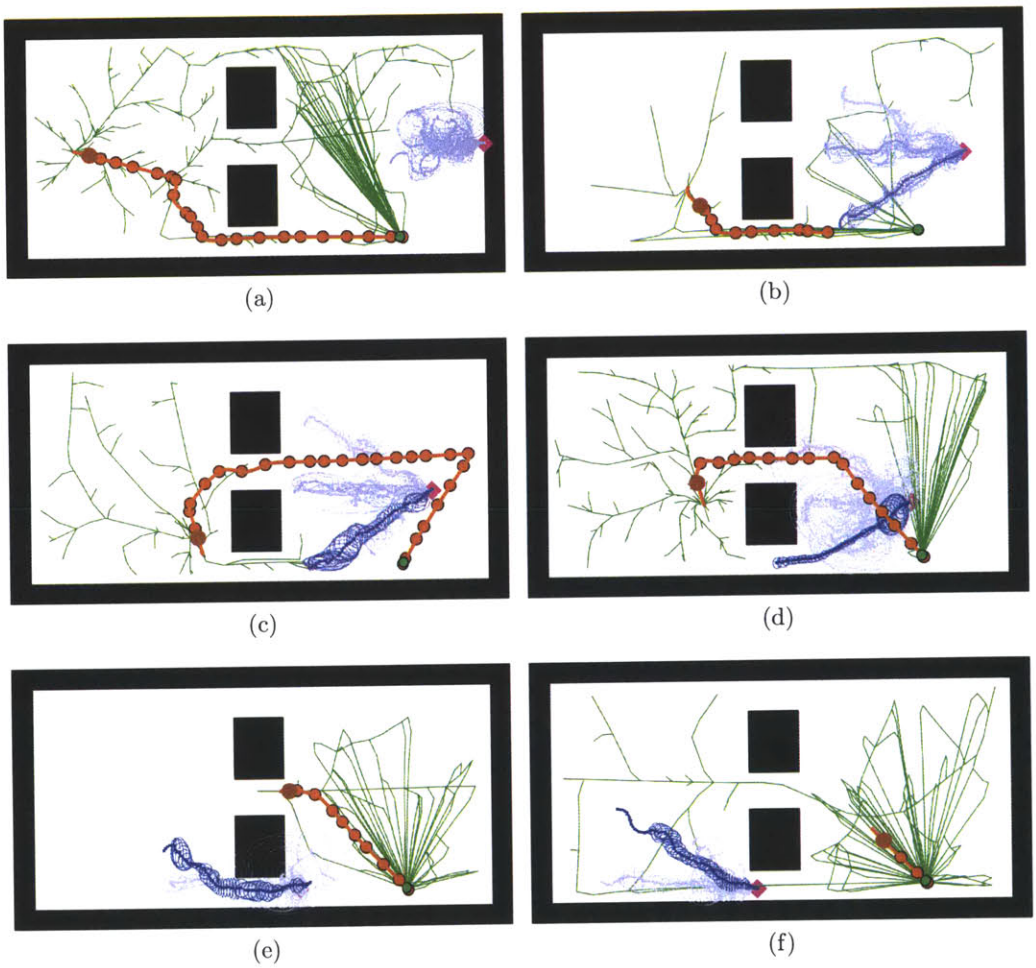


Figure 3-4: Representative snapshots of integrated DPGP and CC-RRT algorithms, obstacle field example #1

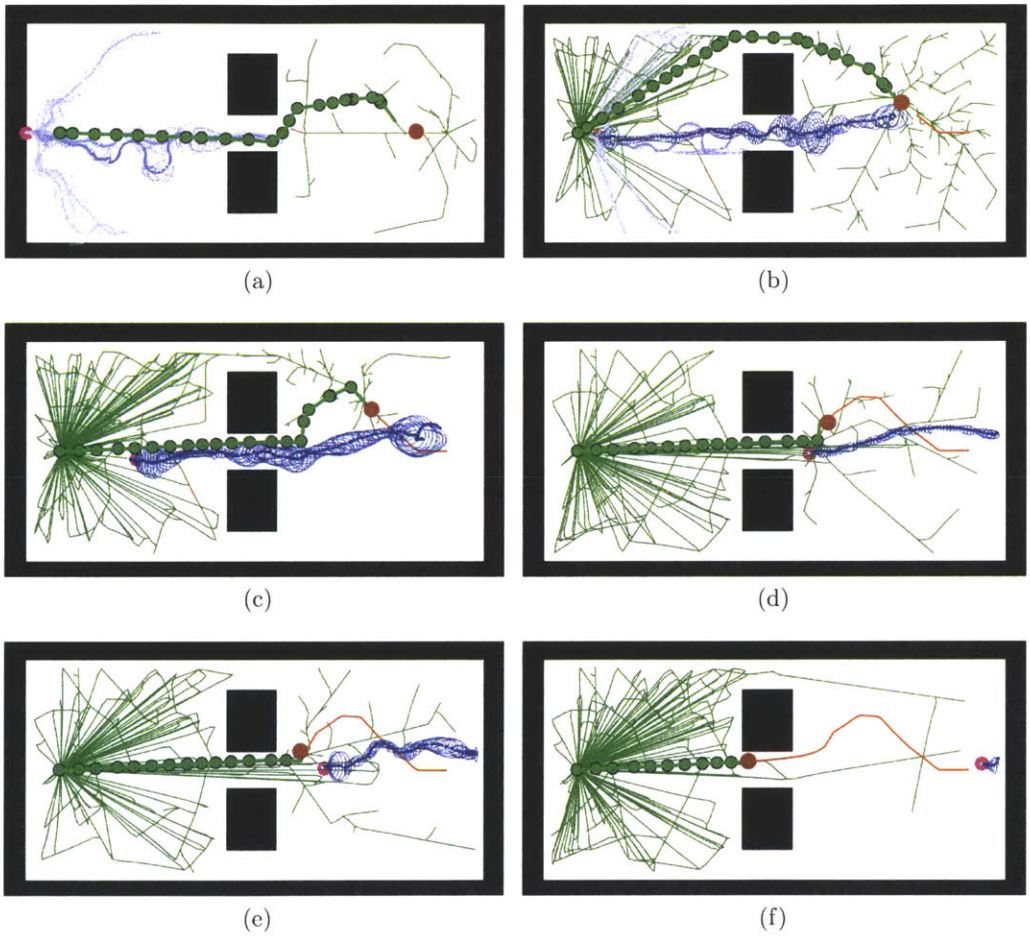


Figure 3-5: Representative snapshots of integrated DPGP and CC-RRT algorithms, obstacle field example #2

Chapter 4

Changepoint-DPGP

This chapter presents an efficient trajectory classification and prediction algorithm that has been developed to enable quick detection of changes in intent and online learning of new behaviors not previously observed in training data, thereby improving predictive accuracy of pedestrians within the environment. The key idea is to employ a novel changepoint detection algorithm [21] for online classification of trajectory segments, decoupling classification and prediction. The resulting approach, Changepoint-DPGP (CP-DPGP), retains the trajectory prediction accuracy of existing GP approaches while expanding their capabilities.

This chapter begins with a problem statement, which motivates and defines the objective of the CP-DPGP algorithm (Section 4.1). The key components of the algorithm, namely changepoint detection (Section 4.2), classification (Section 4.3), and trajectory prediction, are presented next (Section 4.4). Finally, a demonstrative scenario using pedestrian trajectories collected from a 3D lidar illustrates the advantages of the CP-DPGP algorithm by comparing it to two existing trajectory prediction algorithms (Section 4.5).

4.1 Problem Statement

The focus of this work is the accurate prediction of the future location of a moving agent within a specific environment. It is assumed that the agent moves according to

both internal state and environmental features, such that observed trajectories are a representative of both. As such, observed trajectories can be classified into common mobility patterns defined by environmental features (e.g. obstacles, corridors, goal locations), with outliers representative of the internal state of individual agents. Because the environment may change between the training and testing periods and new behaviors may be exhibited online, it is necessary that this set of mobility patterns be updated online to reflect current observations.

The objective is therefore to calculate the future position distribution of agent i for K time steps into the future given this common set of mobility patterns

$$p(x_{t+K}, y_{t+K}|x_{0:t}, y_{0:t}) = \sum_{j=1}^{M_t} \underbrace{p(x_{t+K}, y_{t+K}|x_t, y_t, b_j)}_{\text{position distribution given } b_j} \underbrace{p(b_j|x_{0:t}, y_{0:t})}_{b_j \text{ probability}} \tag{4.1}$$

where M_t is the number of learned motion patterns b_j at time t.

The focus of this work is the prediction of the future position distribution of an agent i moving according to a set of motion patterns. Future trajectories of the agent are estimated given the M_t learned motion patterns at time t $(b_1, ..., b_{M_t})$, with priors $p(b_1), ..., p(b_{M_t})$, where new motion patterns are added as they are learned online.

4.2 Changepoint Detection

To effectively anticipate the motion of pedestrians, a framework is proposed to perform online classification of observed trajectories, in addition to learning common pedestrian trajectories from batch data. For further details, see Grande [21]. Because agile dynamic agents such as pedestrians may exhibit new behaviors or mid-trajectory changes in intent, this problem is framed in the context of changepoint detection. Previous work on changepoint detection includes algorithms such as CUSUM [10], the generalized likelihood test (GLR) [10], and the Bayesian online changepoint detection (BOCPD) algorithm [2].

This work utilizes a variation of the generalized likelihood test (GLR) [10] to perform changepoint detection. The basic GLR algorithm detects changes by comparing

a windowed subset of data to a null hypothesis. If the maximum likelihood statistics of the windowed subset (i.e. mean and standard deviation) differ from the null hypothesis significantly, the algorithm returns that a changepoint has occurred [21].

The changepoint detection algorithm is given in Algorithm 1. At each time step, given Gaussian process GP_w, the algorithm creates a new GP (GP_S) with the same hyperparameters, but using a windowed data subset S of size m_S (lines 2–4). Although m_S is domain specific, the algorithm is fairly robust to its selection; $m_S \approx 10 - 20$ works well for most applications.

The algorithm then calculates the joint likelihood of the set having been generated from the current GP model (the null hypothesis H_0) and the new GP_S (H_1). At each step, the normalized log-likelihood ratio test (LRT) is computed as

$$L(y) = \frac{1}{m_s}(\log P(\mathcal{S} \mid H_1) - \log P(\mathcal{S} \mid H_0)). \tag{4.2}$$

For a GP, the log likelihood of a subset of points can be evaluated in closed form as

$$\log P(y \mid x, \Theta) = -\frac{1}{2}(y - \mu(x))^T \Sigma_{xx}^{-1}(y - \mu(x)) - \log |\Sigma_{xx}|^{1/2} + C, \tag{4.3}$$

where $\mu(x)$ is the mean prediction of the GP and

$$\Sigma_{xx} = K(x, x) + \omega_n^2 I - K(X, x)^T (K(X, X) + \omega_n^2 I)^{-1} K(X, x) \tag{4.4}$$

is the predictive variance of the GP plus the measurement noise. The first term of the log-likelihood accounts for the deviation of points from the mean, while the second accounts for the relative certainty (variance) in the prediction.

Algorithm 1 uses the LRT to determine if the maximum likelihood statistics (mean and variance) of GP_S differ significantly from the null hypothesis, indicating that the points in $\mathcal{S}$ are more unlikely to have been generated from the model GP_w. In particular, the average over the last m LRT values (line 6) is compared to the nominal LRT values seen up until this point (line 7). If the difference of these two values exceeds some value η, the algorithm returns false, indicating that this generating

Algorithm 1 Changepoint Detection [21]

1: **Input**: Set of points $\mathcal{S}$, Working model GP_w
2: $l_1 = \log p(\mathcal{S} \mid GP_w)$
3: Create new GP GP_S from $\mathcal{S}$
4: $l_2 = \log p(\mathcal{S} \mid GP_S)$
5: Calculate LRT $L_i(y) = \frac{1}{m_S}(l_2 - l_1)$
6: Calculate average of last m LRT:
$L_m = \frac{1}{m}\sum_{j=i-m}^{i} L_j(y)$
7: Calculate average of LRT after changepoint:
$L_{ss} = \frac{1}{i-m-1}\sum_{j=1}^{i-m-1} L_j(y)$
8: $i = i + 1$
9: **return** $L_m - L_{ss} \geq \eta$

model does not fit the data.

The value η can be determined based on the probability of false alarms and maximum allowed error [21]. The LRT values tend towards the KL-divergence of the distributions $D(H_0 \| H_1)$. As a designer, one can then decide a maximal error that is acceptable in the model. For the case of two GPs with different means, the KL divergence can be determined in closed form [42]. Roughly, the KL-Divergence between two identical distributions located k standard deviations away from each other is given by $\frac{1}{2}k$. So, for example, to detect new models that are two standard deviations away from the current model, $\eta = \frac{1}{2}(2) \pm \delta$ should be chosen, where $\delta \ll 1$ is some slack factor to account for the fact M may not be large.

The LRT algorithm is quite robust in practice, based on the following intuition. If the points in S are anomalous simply because of output noise, then the new GP model created from these points will on average be similar to the current model. Additionally, the joint likelihood given the new model will not be substantially different from that of the current model. However, if the points are anomalous because they are drawn from a new process, then the resulting GP model will on average be substantially different from the current model, yielding a higher joint likelihood of these points. Lastly, instead of making a decision on a single LRT, the last m LRT's are averaged and compared to the average LRT values seen since the last changepoint. In practice, the LRT may have some offset value due to modeling error. Looking at the difference between the last m values and the average LRT values makes the algorithm

robust to this problem.

4.3 Trajectory Classification

The previous section discussed *changepoint detection*, which must be distinguished from the detection of *changes in intent*. A changepoint refers to the case in which a trajectory segment fits better in a new model than the existing model to which it is being compared; a change in intent refers to a change in the classification of the trajectory segment (i.e. the agent was exhibiting one behavior, then switched to another existing behavior or a new behavior entirely). The two problems are therefore related but distinct, as changepoint detection is necessary to determine changes in intent.

The Changepoint-DPGP algorithm is detailed in Algorithm 2. The algorithm begins with an initial set of learned behavior motion models $\mathcal{GP}$, obtained from running the DPGP algorithm on batch training data. While the changepoint detection algorithm does not require any initial motion models [21], in this application it is desirable to begin with these DPGP clusters as they represent an optimal global classification of the training trajectories. As new data points are received, they are added to a sliding window $\mathcal{S}$ of length m_s. After creating a new model GP_S from the points in $\mathcal{S}$, the LRT is computed for GP_S and for each model GP_j in the current model set $\mathcal{GP}$. This process determines if the points in $\mathcal{S}$ are statistically similar to those in the model GP_j, subject to the predetermined threshold η.

In order to detect changes in intent, the algorithm maintains the set of models $\mathcal{M}_t$ that the points in $\mathcal{S}$ fit into at each time step, representative of the current classification of those points. Because the behavior patterns may overlap (e.g. the blue/green and red/teal behavior patterns in Fig. 4-2b), a single classification cannot be guaranteed, necessitating the maintenance of a model set. Changes in intent occur when the classification changes, i.e. when the current classification $\mathcal{M}_t$ and previous classification $\mathcal{M}_{t-1}$ share no common models. Additionally, the current classification is reset at each timestep to be the intersection of the current and previous classification

Algorithm 2 Changepoint-DPGP

Input: Set of previous behavior models $\mathcal{GP} = \{GP_1, \ldots, GP_N\}$
while Input/Output $\langle x_t, y_t \rangle$ available **do**
  Add $\langle x_t, y_t \rangle$ to $\mathcal{S}$
  Call Algorithm 3
  if $\mathcal{M}_{t-1} \cap \mathcal{M}_t = \varnothing$ **then** {Change in intent detected}
    Reinitialize priors
  end if
  if $\mathcal{M}_t = \varnothing$ **then** {New behavior detected}
    Initialize new model GP_n
  else
    Predict according to Sect. 2.2
  end if
  if $\mathcal{M}_t \neq \varnothing$ **then**
    $\mathcal{M}_t = \mathcal{M}_{t-1} \cap \mathcal{M}_t$
  end if
end while
if GP_n is initialized **then**
  Add $\langle x_{0:T}, y_{0:T} \rangle$ to GP_n
  Add GP_n to set of current models $\mathcal{GP}$
end if

sets, assuming that the current classification is not empty.

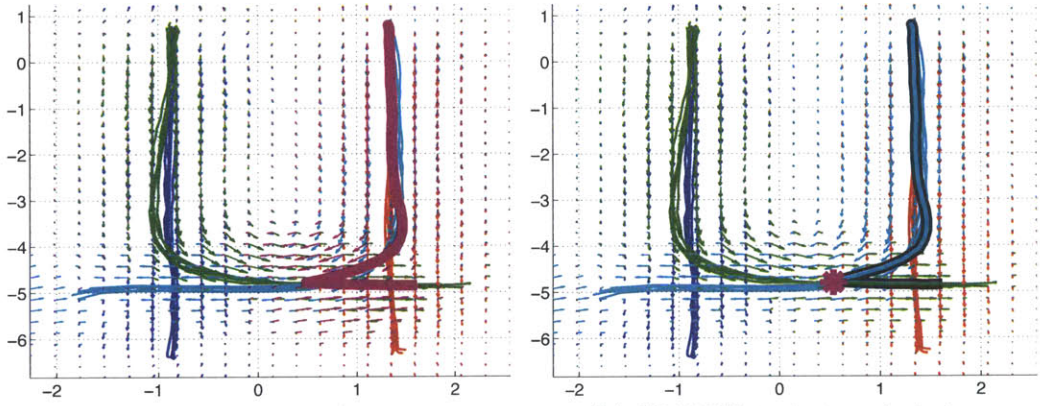
To illustrate this method, consider a pedestrian crossing a crosswalk by following the green behavior pattern in Fig. 4-2b. Until the pedestrian reaches the crosswalk, $\mathcal{M}_t = \{B, G\}$. Once the pedestrian enters the crosswalk, their classification becomes $\mathcal{M}_t = \{G\}$. A change in intent should not be detected at this stage, as the pedestrian is committing to the green behavior rather than selecting a new behavior. However, if the pedestrian switched to the teal behavior after entering the crosswalk, this would represent a change in intent. The classification for three successive timesteps would become $\mathcal{M}_{t-2} = \{G\}$, $\mathcal{M}_{t-1} = \{G, T\}$, $\mathcal{M}_t = \{T\}$ and no change in intent would be detected if $\mathcal{T}_t$ was not reset.

4.4 Trajectory Prediction

The predictive component of this algorithm is decoupled from classification. In general, the future state distribution is computed as described in Section 2.2.3. However,

Algorithm 3 Compare to Current Models

Input: Set of current behavior models $\mathcal{GP} = \{GP_1, \ldots, GP_N\}$
Initialize representative model set $\mathcal{M}_t$
for Each $GP_j \in \mathcal{GP}$ **do**
  Call Algorithm 1 with inputs $\mathcal{S}$, GP_j
  if Algorithm 1 returns **true then**
    Add GP_j to $\mathcal{M}_t$
  end if
end for



(a) DPGP trajectory clustering

(b) CP-DPGP trajectory clustering

Figure 4-1: Trajectory segmentation after change in intent

if at any point $\mathcal{M}_t$ is empty, this indicates that the current model set $\mathcal{GP}$ is not representative for the points in $\mathcal{S}$, so a new behavior must be created. The algorithm waits until the entire new trajectory has been observed to create the new behavior pattern, generating predictions according to a simple velocity propagation model until the model set becomes representative. In practice, any reasonable predictive model can be used at this stage, as no information on the anomalous agent's current behavior is available.

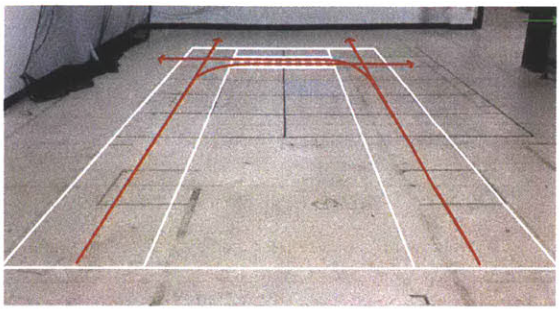
If the online data contains trajectories with changes in intent, the predictive distribution described in Section 2.2.3 will be slow to recognize it, as the prior $p(b_j)$ relies on the entire observed trajectory. Therefore, if a change in intent is detected, the prior probabilities are reinitialized. Likewise, if the training data contains trajectories with changes in intent, DPGP will learn unique behavior patterns for each

trajectory containing such changes, as the entire trajectory is considered for classification. To obtain a representative set of behavior patterns, the Changepoint-DPGP algorithm can be used offline to reclassify these trajectories by segmenting them at the location of the change in intent. To do so, Algorithm 2 is first called with $\mathcal{GP}$ containing those behavior patterns with more than k_{min} trajectories and data $\langle x_t, y_t \rangle$ from trajectories in the remaining behavior patterns not in $\mathcal{GP}$. At line 6 and at the end of Algorithm 2, the trajectory segment seen since the last changepoint is classified into the most likely behavior pattern. The intuition behind these modifications is that changes in intent are agent-specific; therefore, behavior patterns containing these trajectories are not representative of global behaviors caused by the environment.

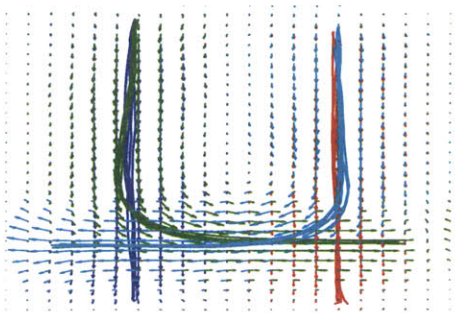
4.5 Simulation Results

This section presents empirical results which evaluate Changepoint-DPGP on the crosswalk scenario in Fig. 4-2, in which pedestrians have four possible behaviors (red) corresponding to which sidewalk they are traversing, and whether they choose to use the crosswalk. The prediction results demonstrate that prior observations of pedestrian motion can be used to learn accurate behavior models. These models are applied to real-time observations to make accurate, long-term predictions of complex motion behavior, beyond what could be predicted from the observations themselves (e.g., bearing and speed).

Three trajectory prediction algorithms are evaluated: Changepoint-DPGP, DPGP, and a goal-directed approach using hidden Markov models (HMM). The hidden states of the HMM are pedestrian goals, learned via Bayesian nonparametric inverse reinforcement learning with an approximation to the action likelihood specifying that pedestrians head directly towards goal locations [36]. Qualitatively, the learned goal locations are the ends of each of the training clusters. This motion model assumes that each pedestrian heads directly toward their intended goal at some preferred speed with an uncertainty distribution over heading and velocity, as used by [22, 8, 25] among others.



(a) Environment for crosswalk experiments, where pedestrian follows one of four possible behaviors (red) and Velodyne location is marked with green arrow

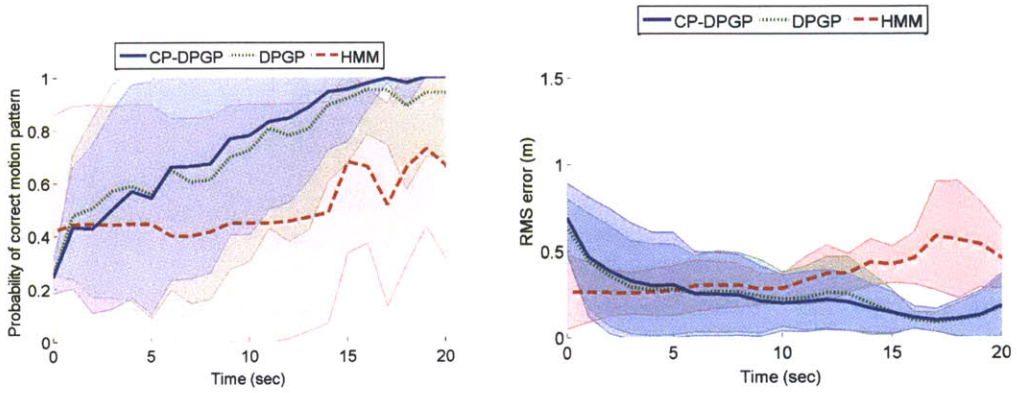


(b) Training pedestrian trajectories collected by Velodyne lidar and resulting DPGP velocity flow fields for each behavior (separated by color)

Figure 4-2: Environment setup and pedestrian data for crosswalk experiments

Unless otherwise noted, all three algorithms were trained on five trajectories from each of the four behavior patterns in Fig. 4-2a. Each trajectory was collected by taking observations of an actual pedestrian traversing the environment, as observed by a Velodyne HDL-32E lidar at the location marked in green in Fig. 4-2a. Pedestrians are identified from the raw Velodyne returns using the algorithm described in Section 5.3.2. Figure 4-2b shows the training trajectories used in this experiment.

Figure 4-3 considers the baseline case in which (i) neither the training nor testing data exhibits any mid-trajectory changes in intent; and (ii) the testing data does not exhibit any behaviors unseen in the training data. In these results, each algorithm is tested on five trajectories from each of the four behavior patterns (Figure 4-2a). Figure 4-3a displays the probability each algorithm is assigned to the correct motion pattern given the observation trajectory, averaged across all 20 trials as a function of time elapsed, with error bars representing standard deviation. This metric measures the ability of each algorithm to identify the correct pedestrian intentions. The likelihoods of each motion pattern serve as the intent prediction for the GP-based approaches, with the prior probability (time = 0) based on the fraction of training trajectories for each motion pattern. The most likely state distribution, calculated via the forward algorithm and Markov chain propagation, describes the predicted intent for the HMM approach. Figure 4-3b displays the root mean square (RMS) error



(a) Probability of correct motion pattern

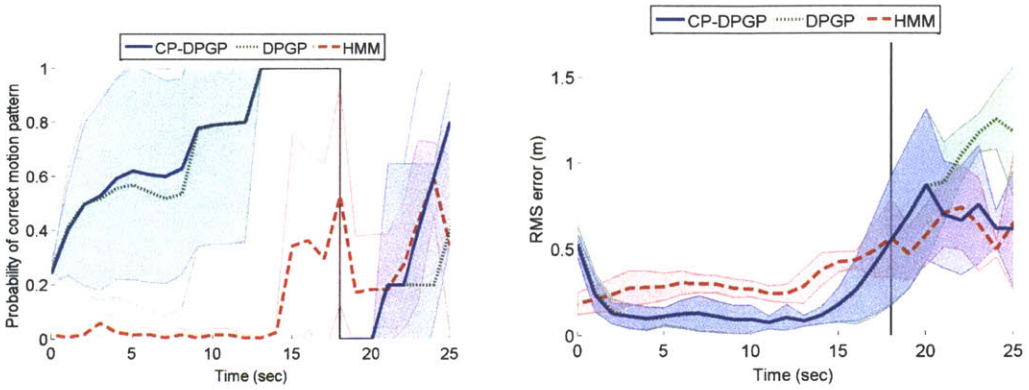
(b) RMS predictive error

Figure 4-3: Comparative prediction accuracy for baseline case of pedestrian crosswalk scenario

between the true pedestrian position and the mean predicted position (Section 2.2), averaged across all 20 trials as a function of time elapsed. This metric captures overall prediction accuracy subject to both intent and path uncertainty.

The Markov property prevents the HMM approach from converging to the correct motion pattern, as the observations of current state alone are not sufficient in the case of noisy observations (Figure 4-3a). As a result, its RMS error tends to increase over time. On the other hand, both GP approaches exhibit convergence in the probability of the correct motion pattern as new observations are made, which improves RMS predictive error as well. The performance of Changepoint-DPGP and DPGP is very similar, as is expected in the absence of changes in intent and new behaviors.

Next, each algorithm is tested on five trajectories which demonstrate a change in pedestrian intentions. In these trajectories, the pedestrian begins to traverse the crosswalk, but reverses direction after 18 seconds. Figure 4-4 shows the evolution of the correct likelihood and RMS error for each algorithm in this scenario, averaged across the trajectories. Both DPGP and Changepoint-DPGP converge on the correct behavior prior to the changepoint (Fig. 4-4a), while HMM performance is relatively unchanged compared to Figure 4-3b. As the change in pedestrian intention takes place, both GP-based algorithms initially drop to zero probability, as expected. How-



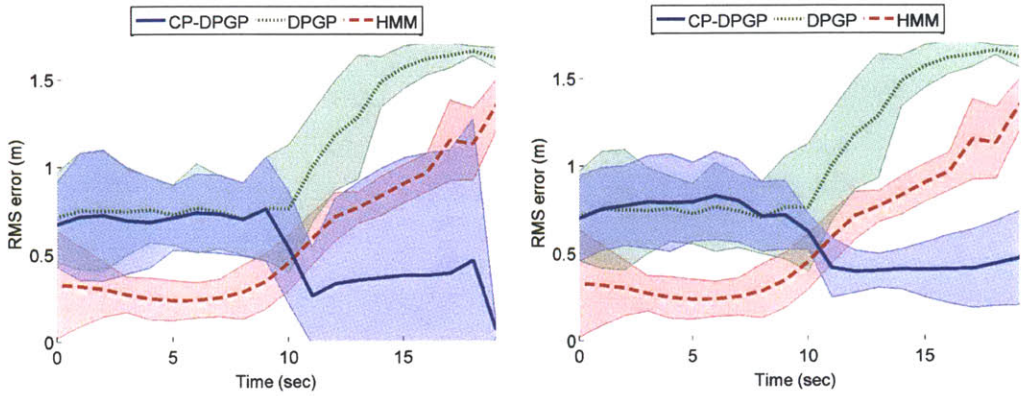
(a) Probability of correct motion pattern

(b) RMS predictive error

Figure 4-4: Comparative prediction accuracy, subject to pedestrian change in intentions at time = 18 s

ever, DPGP accuracy remains poor beyond the changepoint, leading to the largest RMS errors of all algorithms (Figure 4-4b). Because DPGP relies on the entire observation history, its predictions are slow to recognize the change, leading to worse performance. On the other hand, Changepoint-DPGP is able to selectively update the observation history considered in the likelihood computation given changes in intent, enabling it to achieve better accuracy than DPGP after the changepoint (Figure 4-4a). As a result, Changepoint-DPGP yields the lowest average overall RMS error of all algorithms tested (Figure 4-4b).

Changepoint-DPGP also demonstrates the best relative prediction accuracy when considering anomalous/new behavior patterns. In this scenario, algorithms are trained on only three of the four possible behaviors (red, blue, green in Figure 4-2b), then tested on five trajectories from the fourth behavior (teal in Figure 4-2b). The teal behavior deviates from the previously-observed red behavior approximately 9 seconds into the trajectory. Figure 4-5 shows the evolution of the RMS error for each algorithm in this scenario, for the cases when the first predictions of the newly observed trajectory are included and excluded. (Recall that learning occurs at the end of the newly observed trajectory.) At 9 seconds into the experiment, when the pedestrian behavior begins to deviate from anything observed in training data, the



(a) Predictions for first unknown trajectory included

(b) Predictions for first unknown trajectory excluded

Figure 4-5: RMS of predictive error, subject to trajectories not observed in training data

prediction of both HMM and DPGP begins to steadily increase. On the other hand, Changepoint-DPGP successfully identifies the new behavior and reclassifies subsequent trajectories. Thus it exhibits behavior similar to the baseline case, in which predictive error decreases as the probability of the correct motion pattern converges. Overall, Changepoint-DPGP predictive error is reduced by 62% compared to DPGP. This demonstrates the strength of Changepoint-DPGP in cases where the training data is not representative of the observed motion patterns, e.g. due to short periods of data collection or environmental changes.

Chapter 5

Hardware Setup and Data Acquisition

The work presented thus far has been motivated by autonomous navigation in urban environments. Such environments are highly dynamic, often crowded with various agents (e.g. pedestrians, other vehicles), and not presently equipped with sensors capable of providing state information for the autonomous vehicle or other agents. This chapter presents algorithms and sensing infrastructure for control of an autonomous vehicle, in addition to onboard and offboard detection and tracking of other agents, to enable perception-driven navigation. The unifying theme of the hardware components and algorithms presented is that they provide the means for the planning and prediction algorithms discussed in Chapters 3 and 4 to be implemented in the real world.

First, the testbed environment, capable of providing high-fidelity state estimates and real-time display of planning and prediction information, is presented (Section 5.1). Development and control of a small-scale autonomous vehicle is then discussed (Section 5.2), followed by algorithms for detection and tracking of dynamic agents using offboard (3D) and onboard (2D) lidar data (Section 5.3). Because urban environments are populated with many diverse agents, these algorithms are selected to minimize the prior knowledge required of the environment and agents. An extension to a Bayesian nonparametric algorithm [40] is proposed for offline detection and

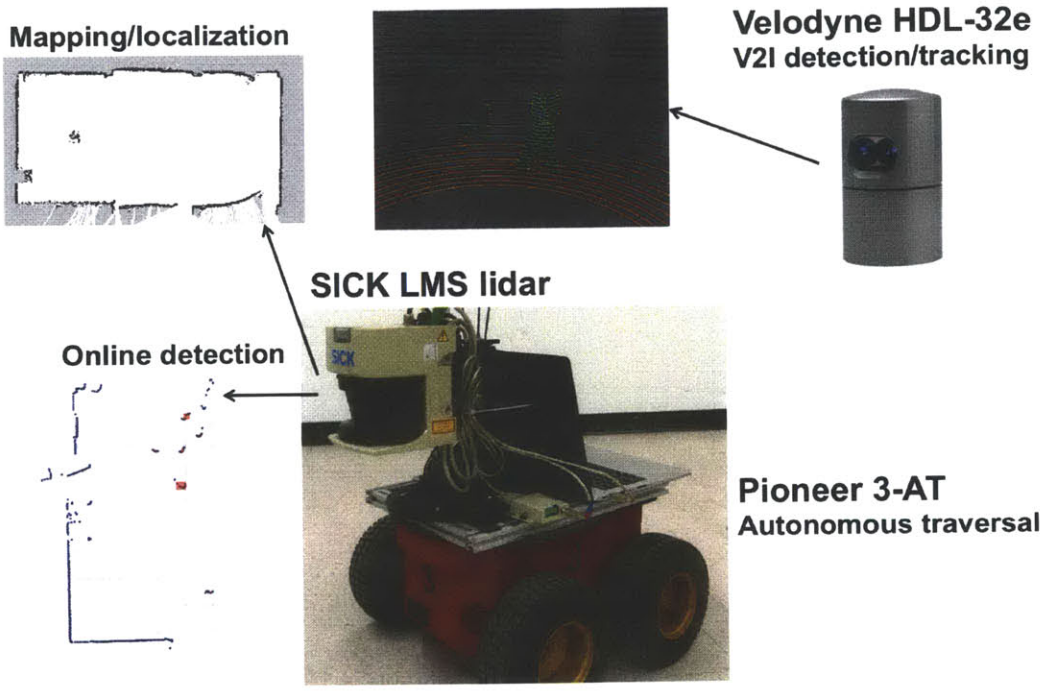


Figure 5-1: Overview of hardware components and corresponding algorithms

tracking without any prior knowledge (Section 5.3.3).

Figure 5-1 provides an overview of the hardware components and algorithms that are discussed in this chapter.

5.1 RAVEN Testbed

Hardware experiments are performed in the Real-time indoor Autonomous Vehicle test ENvironment, or RAVEN [24]. The RAVEN testbed is approximately 12m × 6m in size (Figure 5-2), and contains a set of motion-capture cameras designed to track the location of any operational vehicles [1]. Lightweight reflective markers are attached to vehicles in order to create uniquely detectable configurations. Infrared LED cameras placed around the room (see Figure 5-2a) record the locations where light is reflected. A central processing unit then assimilates each camera's observations to identify the 6 degree-of-freedom (position and attitude) state of each unique marker configuration in the room. This state data is produced and filtered at 100 Hz, with approximately

10 ms delay and sub-millimeter accuracy [51].

A key feature of the RAVEN testbed is the projector array, created by two downward-facing Sony VPL-FH31 projectors [26] that are synchronized to create a contiguous display (Figure 5-2). This display can be used to create artificial environments (e.g. project mountains, rivers, and other natural features) and provide information about the state of vehicles within the room (e.g. exhibit battery status). In this application, it is used to directly overlay planning and prediction algorithm information in the real world, such that the locations of the autonomous vehicle, obstacles, and goal, as well as the motion plan, CC-RRT tree, and predictions, are mapped directly to their physical locations in real-time (Figures 5-2b and 5-2c).

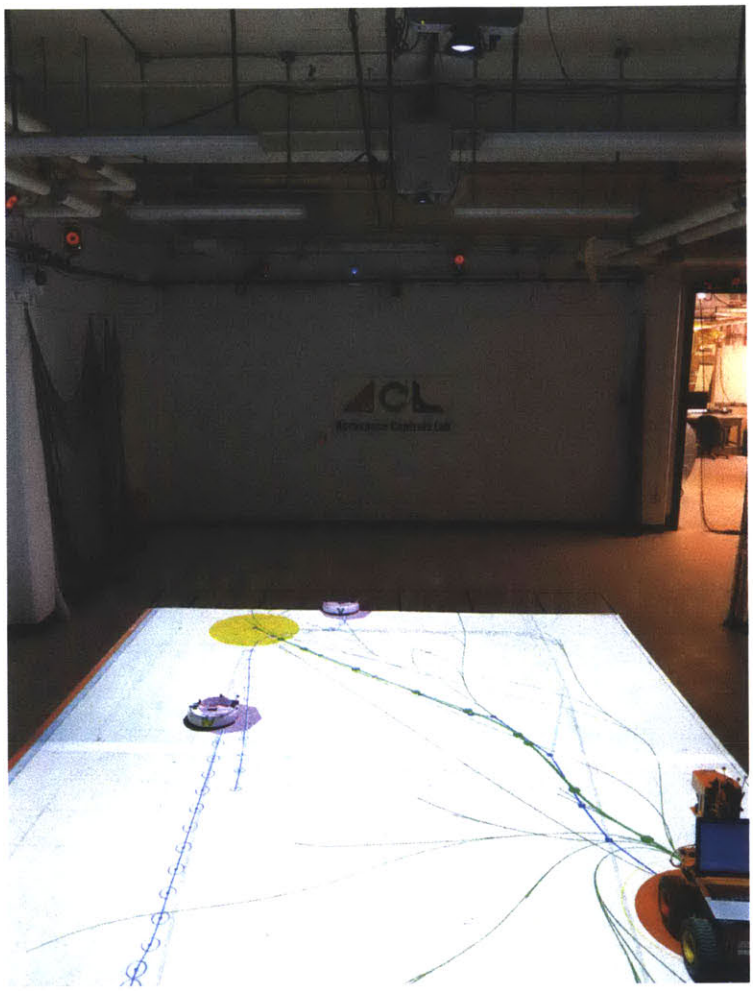
5.2 Autonomous Vehicle

A Pioneer 3-AT rover is used as the autonomous vehicle in all experiments. The Pioneer 3-AT is a four-wheel, four-motor skid-steer rover base, with a maximum speed of 0.7 m/s for each wheel in either direction [38]. Its payload includes a 2D SICK LMS-291 lidar for onboard pedestrian detection and an Intel Core i5 laptop with 6GB RAM for computation (Figure 5-3).

5.2.1 Pure-Pursuit Controller

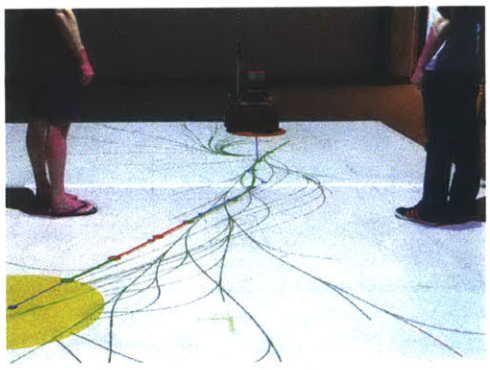
The rover base has internal PID speed controllers that allow for tracking of left and right wheel velocity commands, but an additional controller is required for tracking of a reference path. The pure-pursuit steering controller has been widely used in ground and air vehicle applications, and has the distinct advantage of producing paths that are kino-dynamically feasible by construction [41, 4]. The controller tracks a reference path by adjusting the steering angle of the vehicle to compensate for the error between vehicle heading and the angle between the vehicle and a lookahead point located on the reference path ahead of the vehicle.

The derivation of the pure-pursuit control law for skid-steer vehicle dynamics is described by [17]. The key idea is to modify the standard bicycle model for skid-steer



(a) Representative use of motion capture cameras and projectors.

(b) Projected vehicle path.



(c) Detection of pedestrians.

Figure 5-2: RAVEN testbed.

Figure 5-3: Autonomous rover

dynamics, resulting in the control law

$$\dot{\theta} = -v_{cmd}\left(\frac{\sin\eta}{\frac{L_1}{2}+l_a\cos\eta}\right), \tag{5.1}$$

where $\dot{\theta}$ is the commanded change in vehicle heading angle, v is the velocity control input, l_a is the distance from the front axle to an anchor point aligned with the vehicle heading in the forward direction, L_1 is the distance from anchor point to the lookahead point on the reference path, and η is the angle between the anchor and lookahead points (Figure 5-4).

Note that $\dot{\theta} = \Delta v/L_w$, where L_w is the vehicle track (distance between the left and right wheels) and $\Delta v = v_L - v_R$ is the differential velocity input. Therefore, the control law is related to the system inputs (left (v_L) and right (v_R) wheel velocities)

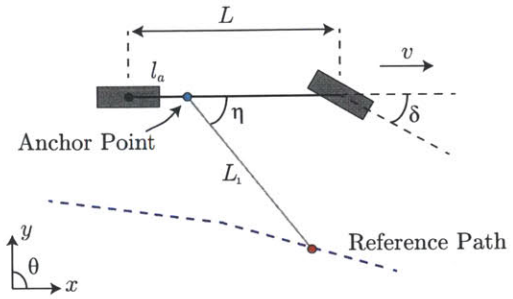


Figure 5-4: Pure pursuit parameters (Source: [17])

by

$$\Delta v = -v_{cmd}\left(\frac{L_w \sin\eta}{\frac{L_1}{2} + l_a \cos\eta}\right), \tag{5.2}$$

where the forward velocity command is the average of the left and right wheel velocity commands ($v_{cmd} = \frac{1}{2}(v_L + v_R)$).

A detailed discussion regarding the selection of L_1 and stability analysis is presented by [31]. It is concluded that the selection of L_1, which determines the position of the lookahead point on the reference path, should be scheduled with speed for improved stability. Namely, larger values of L_1 prevent the vehicle from executing sharp turns, which leads to improved performance and stability at higher speeds. The addition of an anchor point l_a ahead of the vehicle provides additional stability by effectively pulling the vehicle by some point ahead of the axle. In this application, the vehicle is commanded a constant forward velocity of 0.3 m/s; therefore, constant values of 0.4 m for L_1 and 3 cm for l_a are chosen. This values were empirically determined to provide an acceptable tradeoff between trajectory smoothness and vehicle responsiveness.

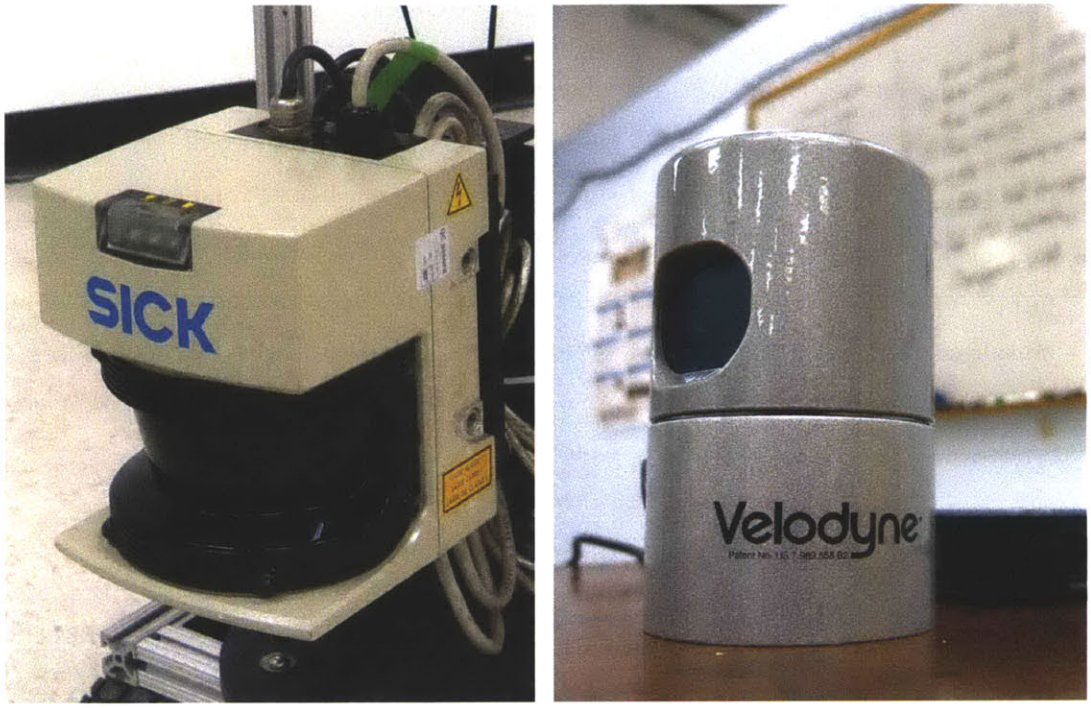
5.3 Dynamic Obstacle Detection

A combination of 2D and 3D laser rangefinders (Section 5.3.1) are used for detection of pedestrians and other agents. These sensors are particularly advantageous because of their high accuracy, long range, and invariance to lighting conditions. Two algorithms have been implemented to cluster lidar returns into objects and track the object centroids. The first is a simple Euclidean clustering algorithm, used for online clustering of 2D and 3D lidar data (Section 5.3.2); the second is an extension of a Bayesian nonparametric algorithm, used for offline clustering of 3D lidar data from crowded environments (Section 5.3.3).

5.3.1 Laser Rangefinders

The 2D SICK LMS 291 laser rangefinder (Figure 5-5a) is mounted on the rover for onboard detection. This lidar scans its surroundings with a radial field of vision using infra-red laser beams. It has a sensing range of 30 meters (at 10% reflectivity, maximum range of 80 meters), with an error of approximately 10 mm. Scans occur at a rate of 75 Hz over a 180 degree range, with 0.25 degree angular resolution [48].

The Velodyne HDL-32E laser rangefinder (Figure 5-5b) enables 3D measurement with a laser scan array of 32 lasers. These beams are organized in multiple planes to provide range, bearing, azimuth, and intensity data of nearby objects. The lasers are aligned from $+10$ to -30 degrees for a 40 degree vertical field of view, and the rotating head delivers a 360 degree horizontal field of view. It has a maximum range of 70 meters and accuracy of ± 2 centimeters [33]. When operating at 10 Hz, this sensor generates upwards of 700,000 points per second. The Velodyne lidar is statically mounted for the collection of training data, and could function as a V2I system by providing real-time estimates. Figure 5-6 portrays a representative sample of 3D lidar data collected on a street, in which point locations represent distance and colors encode intensity. The corresponding camera image is presented on the left.



(a) SICK 2D lidar (b) Velodyne 3D lidar

Figure 5-5: 2D and 3D lidar.

5.3.2 Euclidean Clustering

A difficulty in clustering 2D and 3D lidar returns is extracting useful information in real-time, due to the high scanning frequency and large volume of returns per scan. The Euclidean clustering algorithm as presented in [46, 47] is extremely efficient and can be applied for online clustering of both 2D and 3D lidar returns. Data from these scanners is unorganized, such that the number and ordering of returns may vary across time steps. The Euclidean clustering algorithm employs a kd-tree to iteratively form clusters comprised of nearest neighbors that are within some pre-specified minimum distance, rather than relying on some sort of structure within the measurement data. Algorithm 4 summarizes the relevant clustering steps; this algorithm generates clusters for each new scan in real time.

Performance is highly sensitive to the minimum distance d_k heuristic. Specifically, if this value is too small, a single object can be split into multiple clusters; conversely, a

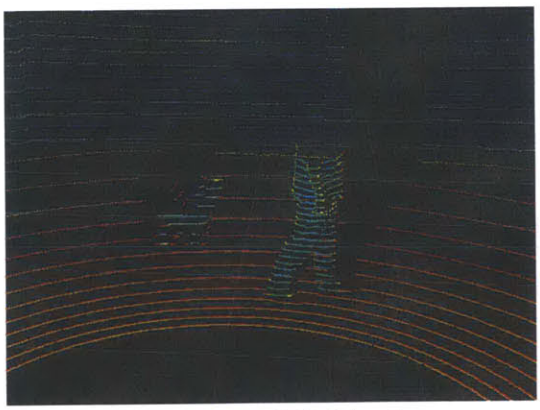
Figure 5-6: Video camera still (left) and 3D lidar data (right) from Vassar Street

value that is too large may cause multiple objects to be merged into a single cluster. This heuristic is therefore object-specific and particularly difficult to determine in environments crowded with objects of varying size and shape. Additionally, because the clusters are recreated at each time step, the algorithm is not robust to occlusion or missed detections. Despite these issues, the algorithm can be tuned to work quite well in practice (Figure 5-7), provided that the objects to be detected are relatively uniform (e.g., pedestrians) and the environment is not overly crowded.

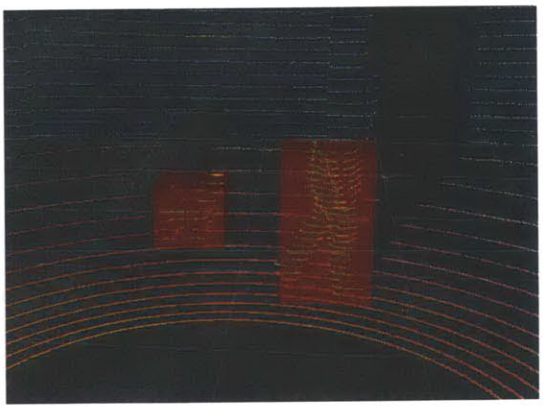
5.3.3 Bayesian Nonparametric Mixture Model

The poor performance of the Euclidean clustering algorithm in crowded environments motivates the need for a detection algorithm that does not rely on specific heuristics. A variety of existing methods have been developed for 3D lidar sensors; however, these methods rely on prior knowledge of the environment and objects to be tracked, which is incorporated into pre-trained and/or heuristic detectors with complex object-specific dynamic models for tracking [39, 43, 7]. This is undesirable for cases in which

(a) Top-down lidar and corresponding forward-facing camera views of 2D clustering



(b) Unclustered 3D lidar data



(c) Clustered 3D lidar data

Figure 5-7: Euclidean clustering results

Algorithm 4 Euclidean Clustering [46]

Input: Point cloud dataset $\mathcal{P}$
Create kd-tree representation for $\mathcal{P}$
Initialize clusters $\mathcal{C} = \varnothing$, queue of points to be checked $\mathcal{Q} = \varnothing$
for all $\mathbf{p}_i \in \mathcal{P}$ **do**
  Add $\mathbf{p}_i$ to current queue $\mathcal{Q}$
  for all $\mathbf{p}_j \in \mathcal{Q}$ **do**
    Query kd-tree for set $\mathcal{P}_j^k$ of point neighbors of $\mathbf{p}_j$ within distance d_k
    Add all neighbors $\mathbf{b}_j^k \in \mathcal{P}_j^k$ that have not been processed to $\mathcal{Q}$
  end for
  Add $\mathcal{Q}$ to $\mathcal{C}$, reset $\mathcal{Q} = \varnothing$
end for

there is incomplete and/or inaccurate prior information about the objects to be encountered. In an urban setting, there is a large number of objects, some of which may appear less frequently (e.g., strollers, pets, wheelchairs) than more common objects (e.g., pedestrians, vehicles). While common objects are likely to appear quickly in training data, infrequently observed objects are unlikely to appear unless a large amount of training data is collected.

A time-dependent Bayesian nonparametric mixture model that does not rely on any prior knowledge of the objects or environment, originally developed for detection and tracking of arbitrary objects from video data [40] is extended for use with 3D lidar data. The algorithm is first summarized, with detail devoted to the changes necessary for lidar data, followed by experimental results from synthetic and real-world experiments.

Generalized Polya Urn Dependent Dirichlet Process Mixture Model

The Generalized Polya Urn Dependent Dirichlet Process Mixture Model (GPUDDPM) allows for the number of clusters (corresponding to the number of objects) at each time step t to be inferred. Figure 5-8 provides a graphical model of the GPUDDPM. Each observation $\boldsymbol{x}_{i,t}$ is associated with an assignment variable $c_{i,t}$ that represents its assignment to cluster $\theta_{k,t}$, where $k \in \{1, ..., K_t\}$ and K_t is the total

number of clusters. The GPUDDPM can be defined generatively as

$$\begin{aligned}
m_{k,t}|m_{k,t-1}, c_{1:N_t,t}, \rho &\sim \mathrm{D}(m_{k,t-1}, c_{1:N_t,t}, \rho) \\
\theta_{k,t}|\theta_{k,t-1} &\sim \begin{cases} P(\theta_{k,t}|\theta_{k,t-1}) \text{ if } k \leq K_t \\ G_0 \text{ if } k = K_{t+1} \end{cases} \\
c_{i,t}|m_{1:K_t,t}, \alpha &\sim \mathrm{C}(m_{1:K_t,t}, \alpha) \\
\boldsymbol{x}_{i,t}|c_{i,t}, \theta 1 : K_t, t &\sim \mathrm{F}(\theta_{c_i,t}, t)
\end{aligned}$$

for all times t and clusters k. The Binomial distribution $\mathrm{D}(m_{k,t-1}, c_{1:N_t,t}, \rho)$ over the size of cluster k at time t (defined as $m_{k,t}$) given the cluster's previous size, current assignments, and deletion parameter ρ, combined with the Categorical distribution $\mathrm{C}(m_{1:K_t,t}, \alpha)$ over the assignment variable $c_{i,t}$ given the current sizes of all clusters and concentration parameter α, comprise the Generalized Polya Urn. The distributions for F, G_0, and $P(\theta_{k,t}|\theta_{k,t-1})$, respectively representing object appearance, appearance prior, and object movement, are application-specific.

The object appearance, appearance prior, and object movement distributions, specified by F, G_0, and $P(\theta_{k,t}|\theta_{k,t-1})$, are the same as those used by [40]. Specifically, the observation $\boldsymbol{x}$ at each time t is represented by a draw from the object appearance distribution, which is the product of a multivariate normal and multinomial distribution. Intuitively, the spatial features of each object are therefore represented by an ellipsoid, whereas the intensity features associated with each object are assumed to be similar. The appearance prior is the product of a normal-inverse-Wishart prior placed on the multivariate normal parameters and a Dirichlet prior placed on the multinomial parameter. The transition kernel representing object movement is kept general to allow for tracking of arbitrary objects, and therefore represented by auxiliary variables.

The Sequential Monte Carlo sampler (particle filter) is implemented to perform inference. The algorithm is fully detailed in [40].

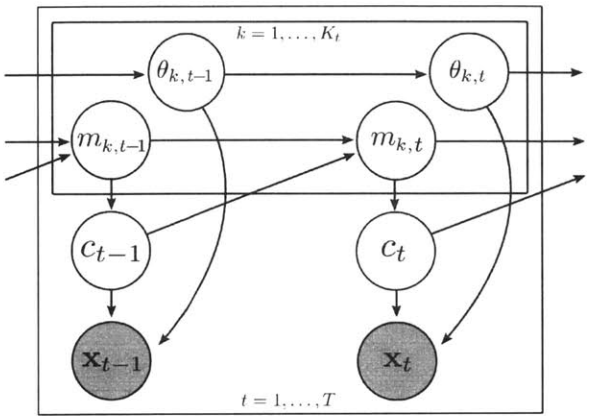


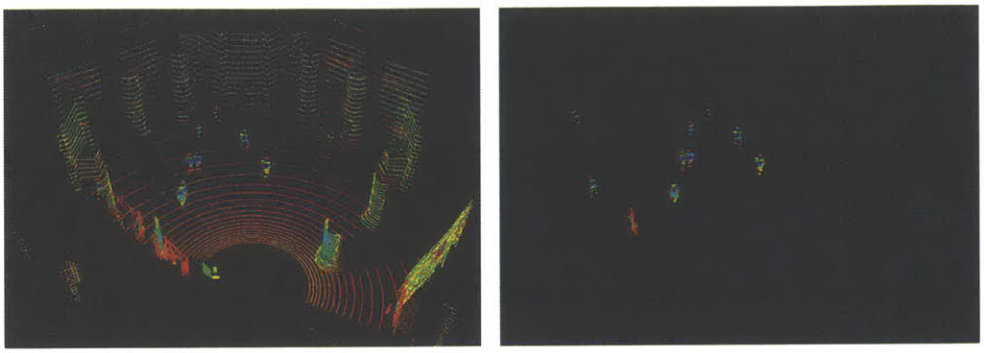
Figure 5-8: Graphical model of the GPUDDPM (Source: [40])

Data Extraction

Due to the high volume of data points at each time step, it is necessary to preprocess the data by determining points of interest. Specifically, points that undergo motion are extracted to yield observations of the form $\boldsymbol{x} = (\boldsymbol{x}^s, \boldsymbol{x}^f, t)$, where $\boldsymbol{x}^s$ is the spatial location of the pixel, $\boldsymbol{x}^f$ is a collection of local image features around the pixel, and t is the time index. For 3D lidar data, $\boldsymbol{x}^s \in \mathbb{R}^3$ and $\boldsymbol{x}^f$ is a V dimensional vector representing the intensity distribution around each point. The set of possible intensity values was partitioned into V bins and $\boldsymbol{x}^f$ contained the counts of the surrounding points in a cube of length L with an intensity value in one of the bins.

The point clouds output by the Velodyne must first be organized before observations can be extracted. Octrees are an efficient method for spatial indexing. The octree recursively subdivides 3D space into eight octants, terminating when the length of the smallest voxel (analogous to a pixel) is below a user-specified threshold. A tree structure in which each internal node has exactly eight children represents this subdivision, where each node contains the indices of data points within the boundaries of the corresponding octree division. The structure of the octrees describing two distinct point clouds can be recursively compared to detect spatial changes that are beyond some threshold [47].

This procedure is somewhat problematic in comparing point clouds that differ by one time step because the changes undergone by moving objects are not sufficient to



(a) Initial point cloud (b) Detection of moving objects

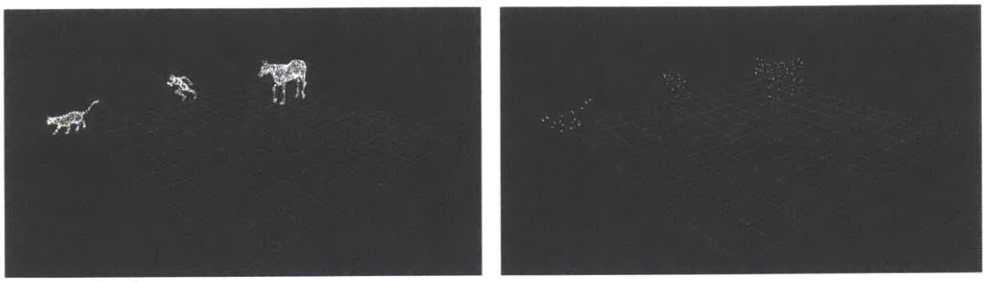
Figure 5-9: Results of octree spatial differencing

overcome false positive detections. Specifically, the laser scanner beams do not fall on the same point in space with each rotation and measurements become sparse with increasing range. The combination of these factors causes stationary objects that are far away (such as the walls bounding the room) to be falsely detected as moving objects unless the spatial difference threshold is sufficiently high. Therefore, spatial changes are detected for point clouds that differ by 2 seconds, such that the spatial difference computed for moving objects is larger than the false detections.

Figure 5-9 displays results of this data extraction procedure for one frame. Moving pedestrians are correctly extracted (Figure 5-9b); however, three stationary pedestrians near the top right corner are not detected because they are stationary (Figure 5-9a).

Results

In order to test the GPUDDPM on 3D lidar data in isolation, a synthetic data set was created, consisting of three objects (man, cat, horse) from the Point Cloud Library data set [47] moving at varied speeds and trajectories for 60 frames. In each frame, the objects were randomly down sampled to approximately 40 points each, as shown in Figure 5-10, in order to more accurately represent the returns from an actual sensor. Note that this test is not entirely realistic for a single sensor, as the point clouds do not occlude each other during movement and the entire surface is detected (rather



(a) Dense (b) Down sampled

Figure 5-10: Dense and down sampled cat (left), man (middle), and horse (right) point clouds

than the partial surface that would be detected by a single sensor). However, it is still useful for testing the model under idealized conditions.

The model is able to accurately classify and track the points belonging to each of the three objects. Figure 5-11 displays representative classification and tracking results for the synthetic point cloud objects (man, cat, horse), where color indicates the current cluster assignment, ellipses represent the covariance in each plane, and the thick black lines indicate the tracked centroid position over time. Tracked centroid positions from the GPUDDPM model (colored) are compared to the actual (black) centroid positions in Figure 5-12.

Trajectories exhibiting sharp corners, such as those executed by the man (blue in Figures 5-11 and 5-12) were the most problematic. This trajectory exhibited the highest average error, 0.428 meters, where the objects were restricted to a cube of approximately $7 \times 7 \times 7$ total meters. As Table 5.1 demonstrates, average error was affected by speed in addition to trajectory smoothness. The slowest-moving object (cat) had the lowest average error in all relevant dimensions (the horse did not translate in the z-direction). Overall, tracking performance was very good, with the largest average error representing just 6.1% of the available space.

This algorithm was also implemented on two crowded data sets from Lobby 7 on MIT's campus collected on different days. Results of a representative classification (Figure 5-13) and tracking (Figure 5-14) are presented. In the tracking results, blue and red lines indicate trajectories collected on a day with (red) and without (blue)

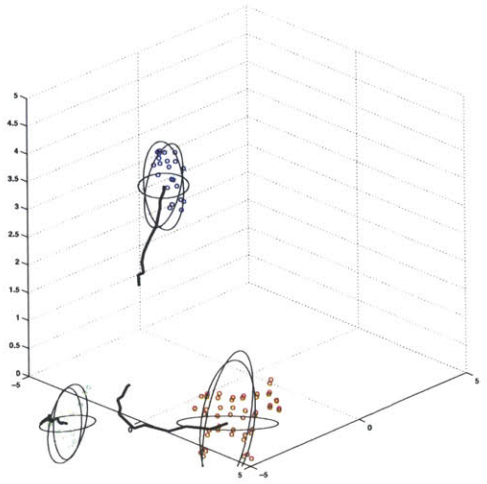


Figure 5-11: Representative classification and tracking results for synthetic point cloud

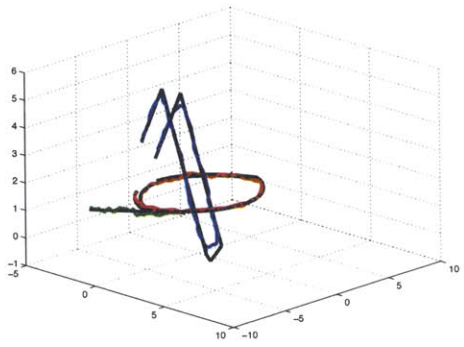


Figure 5-12: GPUDDPM (colored) and actual (black) centroid positions for synthetic point cloud

Table 5.1: Average error in 3D centroid position for each of the synthetic point cloud objects

	x [m]	y [m]	z [m]
Man (blue)	0.428	0.209	0.360
Cat (green)	0.097	0.093	0.057
Horse (red)	0.260	0.398	0.037

construction blocking the center of the space. Because there is no ground truth by which the model estimates can be judged, an empirical discussion replaces accuracy results. In general, the algorithm is observed to be able to successfully track pedestrians in crowded environments; however, problems arise as pedestrians get closer to the laser scanner, casting "shadows" that occlude proportionally larger sections of the environment, or when pedestrians are not moving. The largest drawback of this algorithm is its high time complexity; inference took approximately one hour per minute of point cloud data collected at 5 Hz.

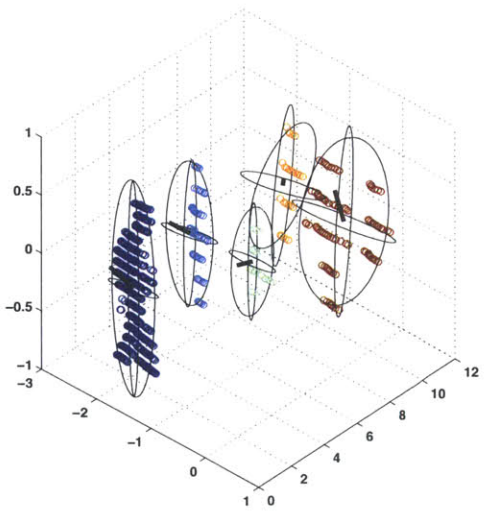


Figure 5-13: Representative classification and tracking results for pedestrians in Lobby 7

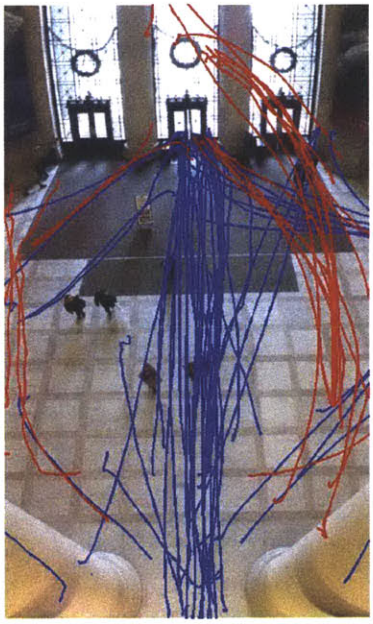


Figure 5-14: Tracking results for pedestrians in Lobby 7

Chapter 6

Experimental Results

This chapter presents experimental results which evaluate Changepoint-DPGP on real-world problem domains of varying complexity. The prediction results demonstrate that prior observations of pedestrian motion can be used to learn behavior models. These models are applied to real-time observations to make accurate, long-term predictions of complex motion behavior, beyond what could be predicted from the observations themselves. The CC-RRT planner is then demonstrated to select safe paths which are risk-aware with respect to possible pedestrian intentions, their likelihood, and their risk of interaction with the host vehicle.

First, the software implementation of the experimental setup is presented (Section 6.1), followed by experimental results. The first set of experiments (Section 6.2), in which the rover plans and executes paths to avoid one or more pedestrians, demonstrates the efficacy of the obstacle detection and vehicle control algorithms discussed in Chapter 5. Predictions of future position are not included; pedestrians are modeled as static obstacles with uncertain locations. In the second set of experiments (Section 6.3), the rover must plan safe trajectories to avoid a pedestrian in a crosswalk environment utilizing Changepoint-DPGP predictions to determine whether or not the pedestrian will cross. The final set of experiments (Section 6.4) involves the rover driving around multiple small dynamic robots whose behaviors are known or learned online.

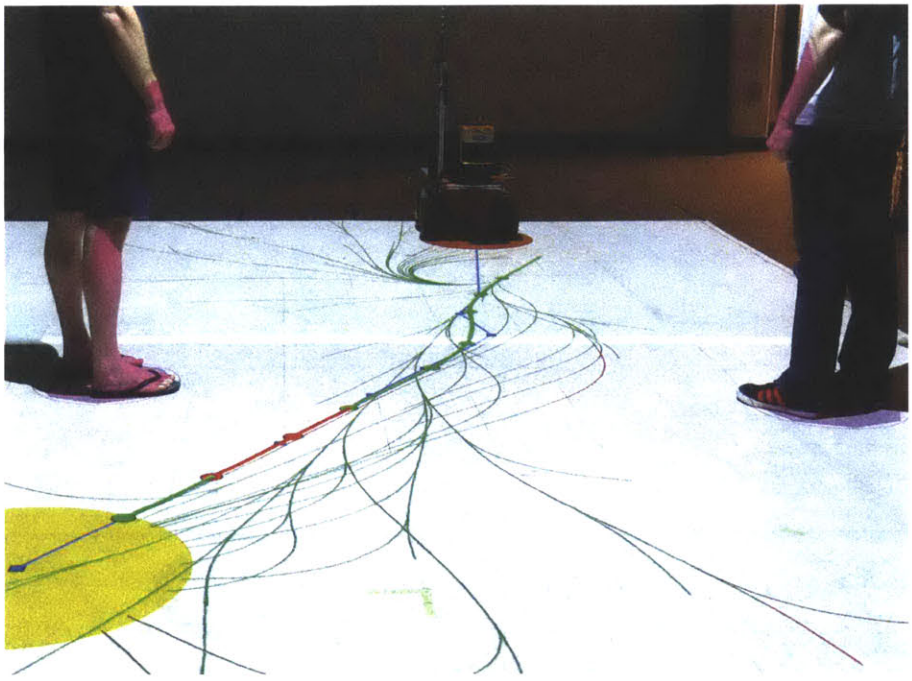
6.1 Implementation

The online planning and control algorithms have been implemented in a multi-threaded, real-time Java application. The software implementation consists of four primary modules, each in a separate thread. The vehicle thread updates the position of the vehicle both in simulation and real-world experiments; it is run in real-time, and operates continuously until a collision has occurred or the vehicle has safely reached its final goal. The CC-RRT thread grows the tree and sends the current best path to the vehicle thread at a fixed rate. A unique thread is launched for each obstacle, which updates the state both in simulation and real-world experiments, in addition to maintaining predictions of the likelihoods and future state distributions of possible behaviors where appropriate. Finally, the communications thread receives vehicle and obstacle state data and broadcasts the path waypoints via WiFi, utilizing the Robotic Operating System (ROS) [44].

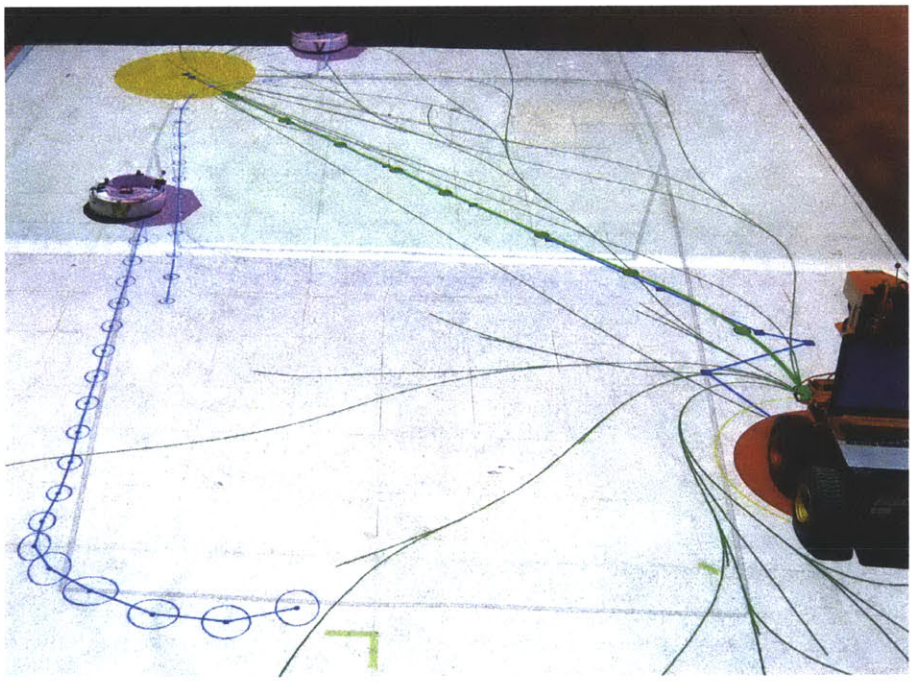
The rover itself has an onboard laptop with Intel Core i5 processor and 6 GB RAM. The laptop receives path waypoints via WiFi and communicates with the rover to send commands and receive state updates via a serial connection. It is also connected to the SICK lidar via a high-speed serial connection, enabling the detection of dynamic obstacles as described in Section 5.3, which are sent to the appropriate obstacle threads via wireless.

To demonstrate the planning and prediction algorithms, a visual overlay maps the CC-RRT and Changepoint-DPGP situational awareness directly onto the physical environment using an array of overhead projectors (Section 5.1); representative visualizations for two scenarios are presented in Figure 6-1. In these figures, and all subsequent experiments discussed in this chapter, the white region on the overlay represents feasible portions of the environment for the rover. However, this region does not represent the *entire* feasible space; the projectors span the full width of the feasible space, but not its length. Therefore, the rover may have feasible paths that take it outside of the projected overlay to reach goal waypoints.

Figure 6-1 shows representative CC-RRT trees grown for the rover to reach a goal



(a) Static robots



(b) Dynamic robots

Figure 6-1: Planning and prediction algorithm visualization for static and dynamic robot scenarios

waypoint while avoiding two static (Figure 6-1a) and dynamic (Figure 6-1b) robots. In the overlay, the rover is represented by a large orange circle, while each robot is represented as a magenta octagon. In the experiments throughout this chapter, both pedestrian and robot obstacles are represented by the same magenta polygon. The goal region for the rover is marked in yellow. The CC-RRT tree (thin edges) and selected path for execution (thicker edges, with circular nodes) are nominally colored green; however, the color is shaded from green to red to indicate higher-risk segments of both tree and path. Dynamic obstacle predictive distributions of future position (square nodes, with 2 σ uncertainty ellipses) are shown in blue, with darker shades indicative of higher likelihoods (Figure 6-1b).

6.2 Static Pedestrians

The first set of experiments serves as an introduction to the testbed environment and performance of the hardware components, data acquisition algorithms, and CC-RRT planner in isolation. These experiments motivate the need for predictions of the future behavior of dynamic obstacles in an uncertain world.

6.2.1 Setup

In these experiments, the autonomous rover must safely navigate around one or more pedestrians to reach a sequence of goal waypoints (Figure 6-1a). Rather than waiting for the rover to exactly reach each goal, the planner switches to the next goal once the rover is within one meter of the goal. Each pedestrian is detected and tracked by the onboard 2D lidar as a potential obstacle. Detections are mapped to the global environment using the location of the rover as determined by the motion-capture system. The CC-RRT planner treats each pedestrian as a static obstacle with uncertain placement due to sensor noise. If a pedestrian drops out of the lidar field-of-view, the obstacle associated with that pedestrian persists for several seconds before disappearing.

Table 6.1: Summary of hardware experiment videos

#	Filename	Description	Section
1	`video1.mov`	2 static pedestrians	6.2
2	`video2.mov`	1 pedestrian traversing crosswalk	6.3
3	`video3.mov`	1 pedestrian remaining on sidewalk	6.3
4	`video4.mov`	1 dynamic robot, baseline scenario	6.4
5	`video5.mov`	2 dynamic robots, baseline scenario	6.4
6	`video6.mov`	1 dynamic robot, online learning	6.4

All videos are located at `http://acl.mit.edu/videos/ferguson-sm/video#.mov` – insert # from table above. Copies were also included in the submission of this thesis to MIT.

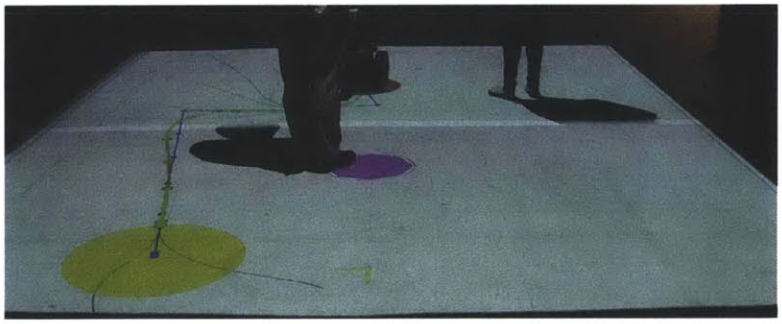
6.2.2 Results

In Video 1 (Table 6.1), the rover safely navigates through a sequence of 6 goal waypoints while avoiding 2 pedestrians; image stills from a relevant interaction in Video 1 are presented in Figure 6-2. Here, the planner identifies a path to the next goal, but is unaware of a pedestrian standing on that path, as the pedestrian is not within the lidar field of view (Figure 6-2a). As the rover turns, this pedestrian is detected by the lidar, whereas the original pedestrian is lost (Figure 6-2b). The original path is ruled probabilistically infeasible and the rover identifies an alternative path (Figure 6-2c), which it follows to safely reach the goal (Figure 6-2d).

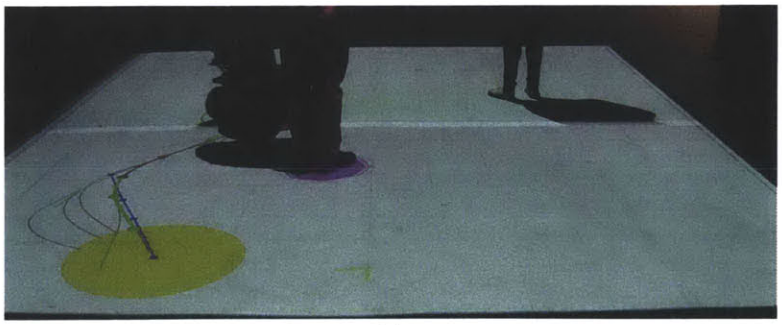
This scenario motivates the need for predictive models of future behavior for dynamic obstacles in uncertain, real-world environments. Sensors have limited capabilities (e.g. the onboard lidar has a limited field of view), so the planner's obstacle data may become out-of-date or miss pedestrians entirely. As such, the planner may generate paths that would collide with pedestrians because it cannot see them; fast (1 second) tree updates and replanning are therefore essential for safe navigation. Planning for uncertain *future* positions, rather than uncertain current location as in this experiment, reduces the risk of collision with dynamic obstacles [6].

(a)

(b)



(c)



(d)

Figure 6-2: Moving rover planning paths around 2 pedestrians

6.3 Pedestrian Crosswalk

The second set of experiments demonstrates the use of predictive distributions given onboard sensing of obstacles. To mimic a real-world scenario, training and testing data is collected from two separate sensors for two different pedestrians with unique walking speeds and gaits.

6.3.1 Setup

Recall the scenario described in Section 4.5, in which pedestrians at a crosswalk have four possible behaviors, corresponding to which sidewalk they are traversing, and whether they choose to use the crosswalk. In this experiment, the rover must safely travel along the street flanked by the two sidewalks and pass through the crosswalk. To do so, the planner uses the same training trajectories presented in Section 4.5 to generate predictions for a live pedestrian as detected by the moving rover. Training trajectories are collected from a stationary 3D Velodyne lidar; online testing trajectories are generated from an onboard 2D lidar and mapped into the global frame using the location of the rover as determined by the motion-capture system. Two different individuals serve as pedestrians in the training and testing data.

6.3.2 Results

In Video 2 (Table 6.1), the rover safely avoids a pedestrian to navigate through a crosswalk. Figure 6-3 provides a time series progression of this scenario. Initially, the vehicle plans a path directly to the goal, as the future position of the pedestrian is not projected to interfere (Figure 6-3a). At the first time step, all behaviors are equally likely (Figure 6-3a); however, the probability of the behaviors on the opposite side of the street correctly converge to zero within one time step (Figure 6-3b). As the likelihood of the crossing behavior converges, the planner terminates the path before the projected collision point (Figure 6-3c) allowing the pedestrian to safely cross before continuing on to the goal (Figure 6-3d).

Video 3 (Table 6.1) and Figure 6-4 proceed in a similar fashion. Because the pedestrian starts slightly ahead of the previous scenario, the vehicle initially plans a path that terminates before the crosswalk, allowing for the possibility that the pedestrian may cross (Figure 6-4a). Due to the uncertainty in whether or not the pedestrian will cross, the planner keeps this shortened path to enable the rover to safely react if the pedestrian does enter the crosswalk (Figure 6-4b) until the timing is such that the pedestrian and vehicle will not collide (Figure 6-4c). As the pedestrian passes by the crosswalk, the motion pattern probability converges to the straight behavior and the vehicle is able to safely reach its goal (Figure 6-4d).

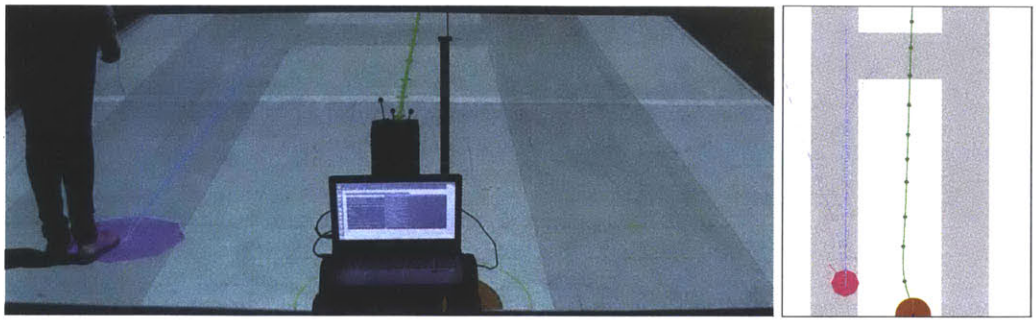
6.4 Dynamic Robots with Uncertain Intentions

The final set of results demonstrates both the efficiency and online learning capabilities of Changepoint-DPGP. Predicted future distributions are generated for multiple dynamic robots in real time, enabling the rover to safely operate in a constrained environment and avoid several other dynamic agents (Section 6.4.3). Previously unobserved behaviors are learned online, enabling robust avoidance where simple velocity propagation methods fail. Together, these results advance the state of the art capabilities of existing algorithms.

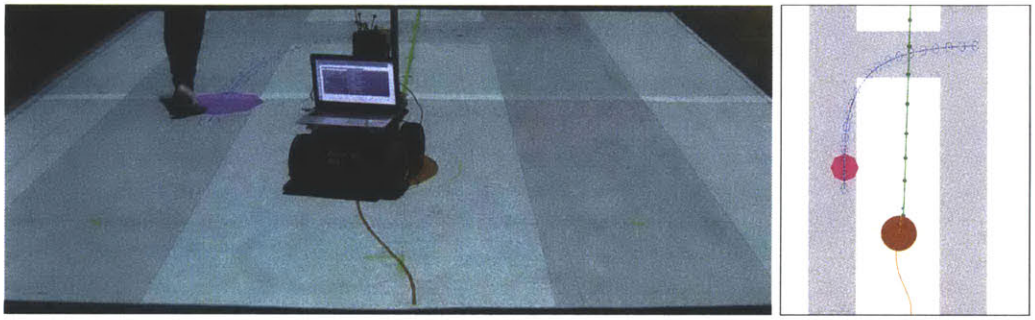
6.4.1 Setup

In these experiments, the autonomous rover must safely navigate around one or more small iRobot Create vehicles [27] ("robots") to reach a sequence of goal waypoints. The planner provides the rover with a fixed sequence of goal waypoints to reach, one goal at a time, located at the four corners of the testing environment. Each robot is detected and tracked by the high-fidelity motion capture system. The robots travel at a forward speed of 0.2 m/s; the rover travels slightly faster at 0.3 m/s.

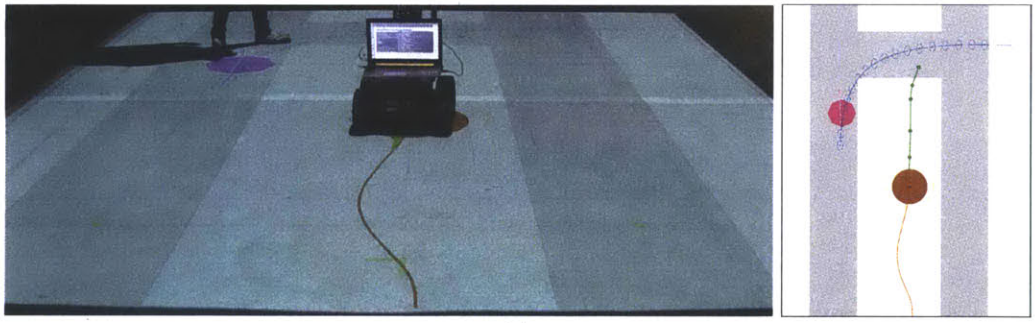
The robots exhibit one of three possible behavior patterns (Figure 6-5). These behaviors are clustered by DPGP from 15 trajectories (5 per behavior) collected offline. Once the robot reaches the darkened area behind the projector display, it is



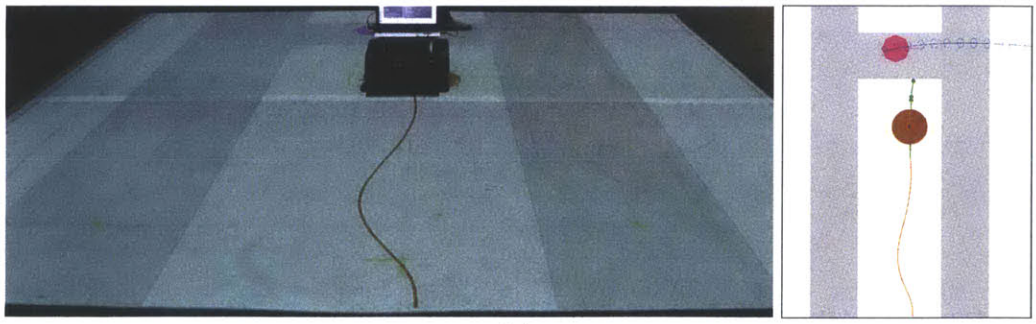
(a)



(b)

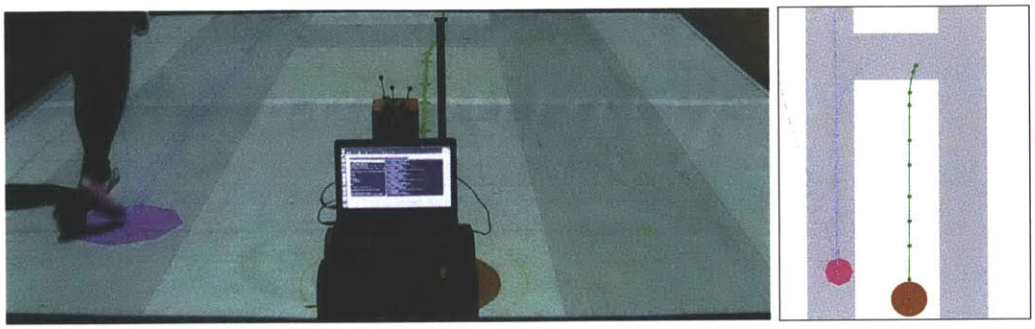


(c)

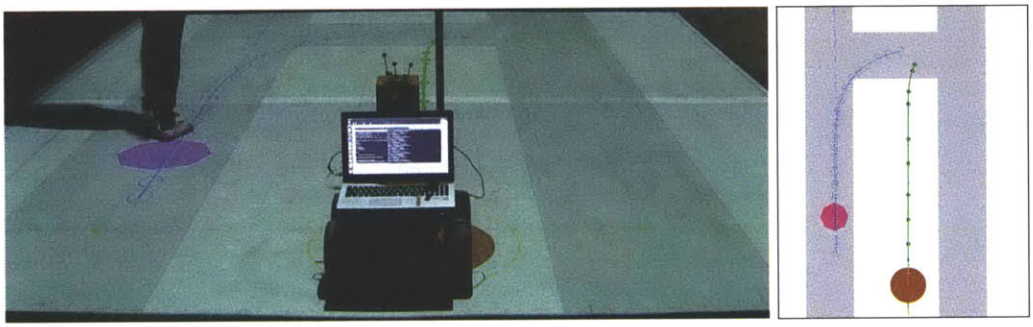


(d)

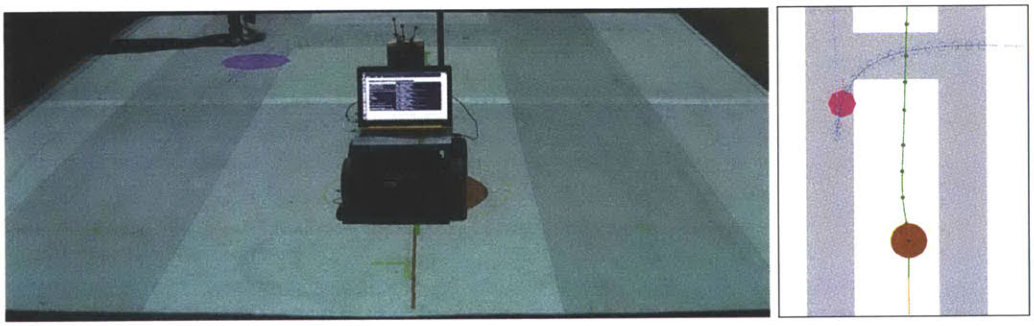
Figure 6-3: Rover avoiding pedestrian traversing crosswalk



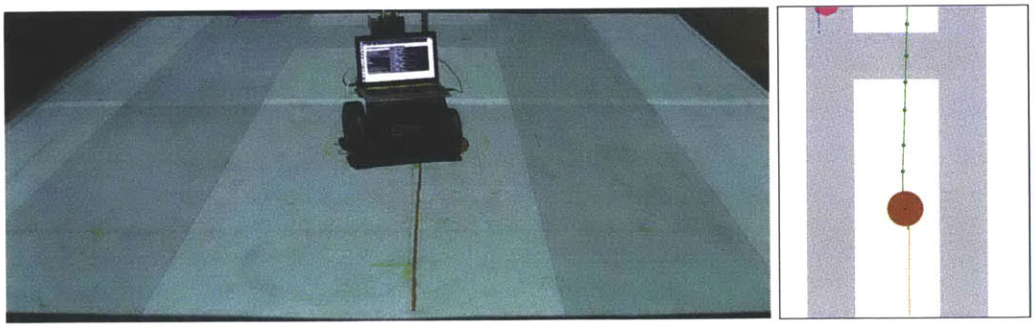
(a)



(b)



(c)



(d)

Figure 6-4: Rover avoiding pedestrian on sidewalk

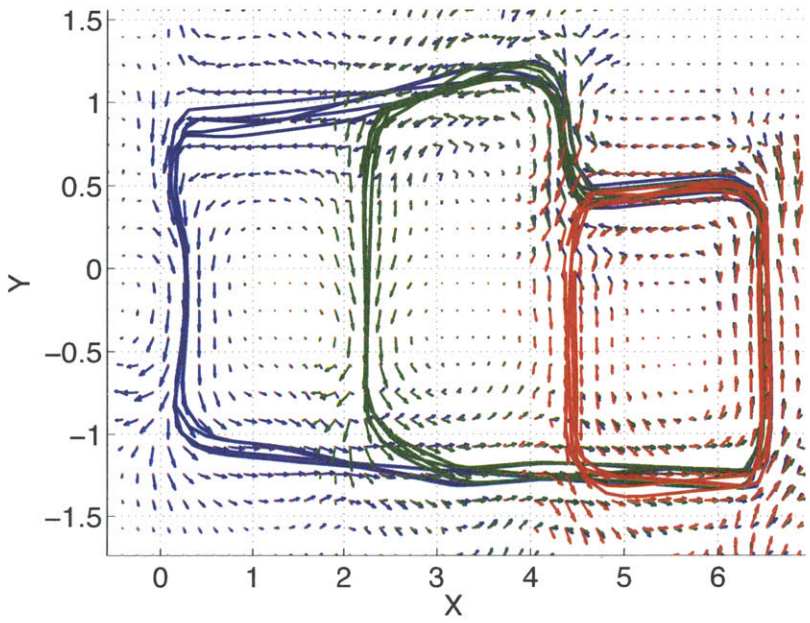


Figure 6-5: DPGP clustering of robot behaviors

sent a new set of waypoints. As these waypoints may comprise an alternate behavior pattern, the likelihoods are always reset once the robot receives a new trajectory. To introduce additional uncertainty in the robot position, 10% noise is added to each waypoint sent to the robots during both the offline training and online testing phases. In the projected display, gray lines indicate the nominal robot trajectories (e.g., Figure 6-6).

6.4.2 Baseline Results

In the baseline scenario, a single robot exhibits the three behavior patterns in Figure 6-5, all of which are known. In Video 4 (Table 6.1), the rover safely navigates through a sequence of 16 goal waypoints while avoiding this robot. Figure 6-6 provides an example of an interesting rover/robot interaction. Initially, the planned path does not reach the goal, due to an anticipated overlap with one of the dynamic robot's possible behaviors (Figure 6-6a). The planner finds a feasible path to the goal, during which time the predictions of the robot's have converged, making this path risky (Figure

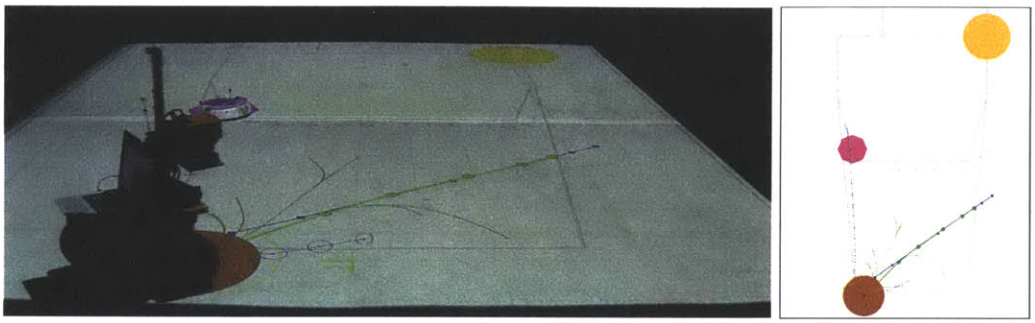
6-6b). The planner prunes this node and the rover comes to a stop and waits for the robot to pass (Figure 6-6c); once the robot has done so, the planner identifies a new path reaching the goal (Figure 6-6d).

6.4.3 Multi-Robot Results

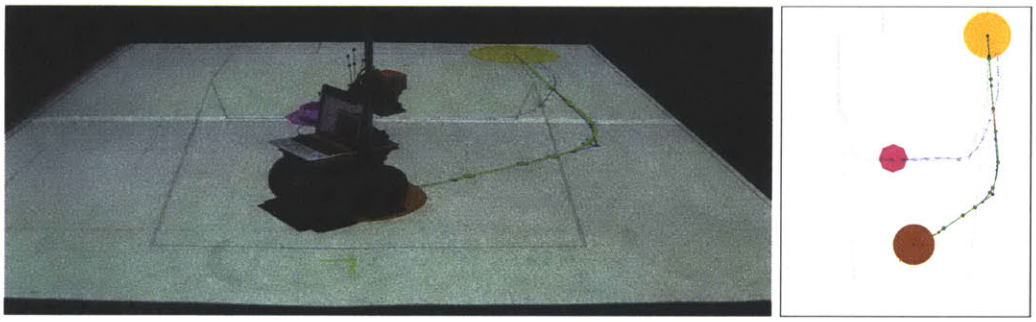
In the multi-robot scenario, two robots exhibit the three behavior patterns in Figure 6-5, all of which are known. In Video 5 (Table 6.1), the rover safely navigates through a sequence of 10 goal waypoints while avoiding both dynamic robots. Figure 6-7 provides an example of a particularly interesting interaction, in which predictions are utilized to enable the rover to safely reach the goal. The goal is directly behind the rover (Figure 6-7a), but the initial planned path diverts far to the right side (Figure 6-7b). In doing so, the planner avoids both possible future predicted behaviors of the visible robot, as the two remaining behaviors are approximately equally likely. As the robot commits to the outermost behavior (blue in Figure 6-5), the current planned path becomes too risky (Figure 6-7c) and the planner identifies a more direct route to the goal (Figure 6-7d).

As the second robot comes into view, its predicted behavior distribution interferes with the rover's path, so the planner updates the path to lead the rover to a safe position within the environment (Figure 6-8e). The planner then finds a new path to the goal (Figure 6-8f), which becomes unsafe as the robot commits to the middle behavior (green in Figure 6-5), forcing the removal of risky path nodes (Figure 6-8g) and generation of a more conservative plan (Figure 6-8h). As this interaction demonstrates, accurate predictions of both intent and trajectory are necessary for safe navigation, as the planner must incorporate a distribution over all possible future behaviors to have sufficient planning and reaction time.

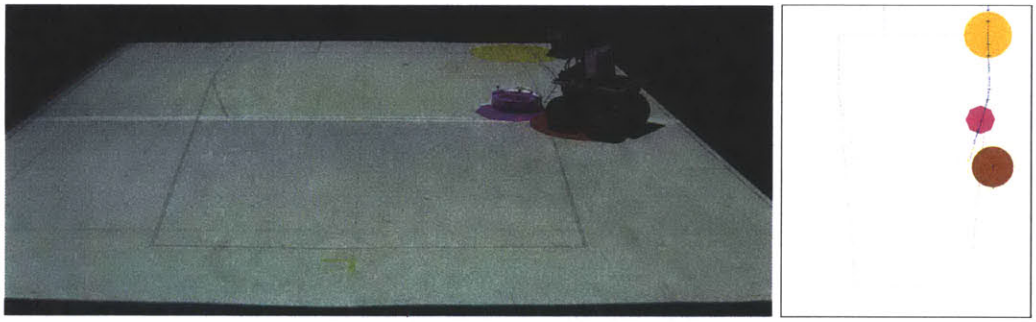
This section concludes with a remark on computational complexity. Because the predictive distribution for each obstacle is dependent only on the current position and learned behavior models, and each obstacle is run as a separate thread, the predictions are efficiently parallelized. In theory, there are no limitations from a predictive standpoint on the number of obstacles considered; in practice, computational



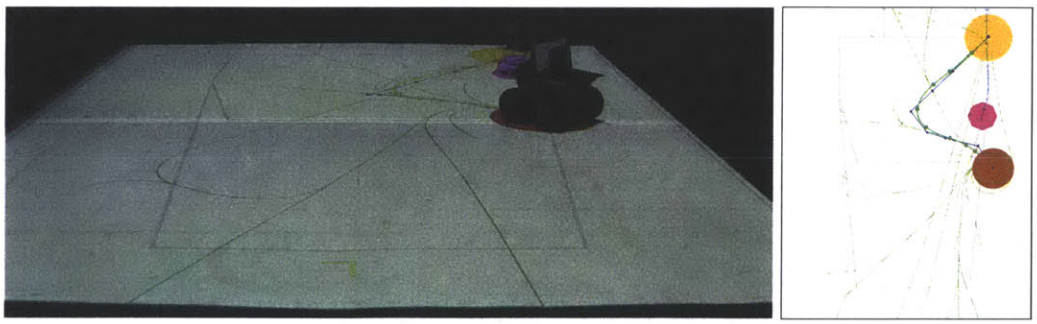
(a)



(b)



(c)



(d)

Figure 6-6: Moving rover planning paths around 1 dynamic robot

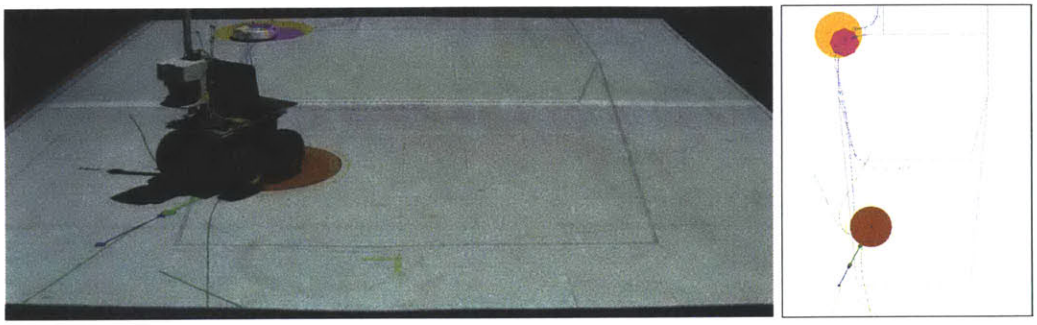
resources may be a limiting factor. The CC-RRT algorithm considers each behavior of each robot as an obstacle, so from a planning standpoint the scaling is linear in both the number of dynamic agents and behaviors [6].

6.4.4 Online Learning Results

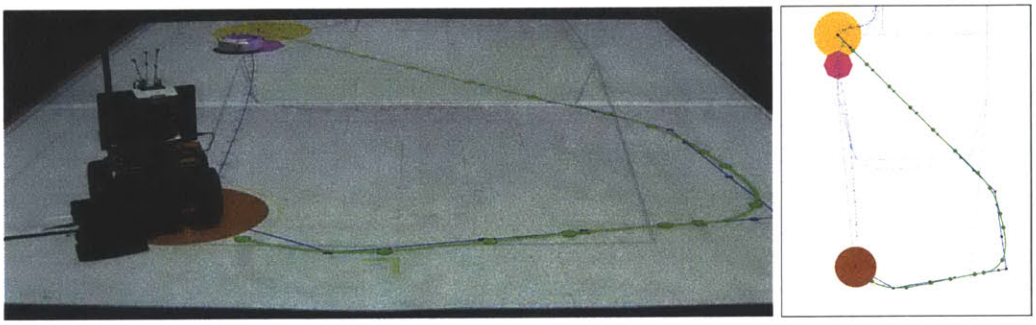
In the online learning scenario, the robot executes the inner- and outermost behaviors (red and blue in Figure 6-5, respectively), but only the innermost trajectories are provided as training data. This section begins with an illustrative example of the Changepoint-DPGP algorithm, followed by a comparison of the predictive error for the known and learned behaviors.

Fig. 6-8 illustrates how Changepoint-DPGP behaves when a new behavior pattern is observed repeatedly. (To avoid obscuring the predictions, the planning tree is not visualized in this figure, and the planned and executed rover paths are shown in dark and light orange, respectively.) Initially, the only known behavior is the innermost cycle (denoted by 1), so the autonomous rover is certain that the robot will pass between it and the goal and modifies its path accordingly. At 8 seconds, Changepoint-DPGP recognizes that the robot is executing a new behavior. Predictions are then generated assuming that the robot will continue at its current velocity with increased, linearly-scaling uncertainty. The planner modifies its planned paths to the goal to reflect this shift in perceived behavior at 25 and 36 seconds.

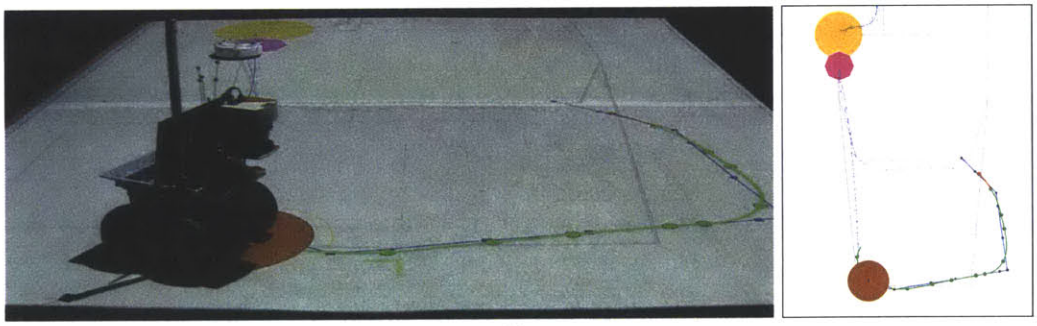
After 92 seconds, the algorithm has learned the entire trajectory that it has just observed as a new behavior. As the robot begins its second cycle, it still assigns the highest likelihood to the known behavior (behavior 1), based on the prior distribution of observed training and test trajectories still favoring this behavior. However, the new behavior is now included as an additional behavior prediction. By 97 seconds, the algorithm is fully confident that the robot is executing the newly-learned, and shifts its likelihoods accordingly. The predictions for the new behavior accurately reflect the trajectory executed by the robot with reduced uncertainty. Based on this reduced uncertainty, the planner knows that the robot will turn before intersecting with the autonomous rover's planned path, and thus continues to execute that path.



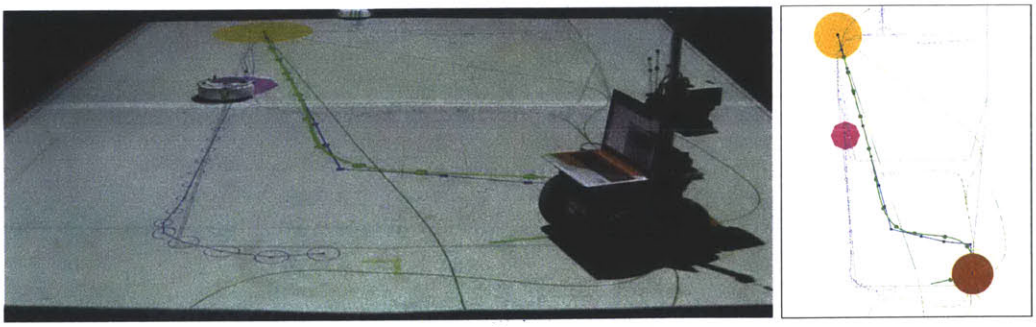
(a)



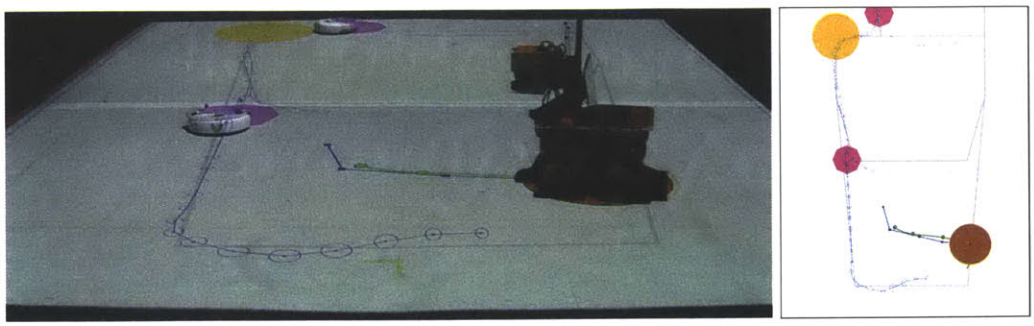
(b)



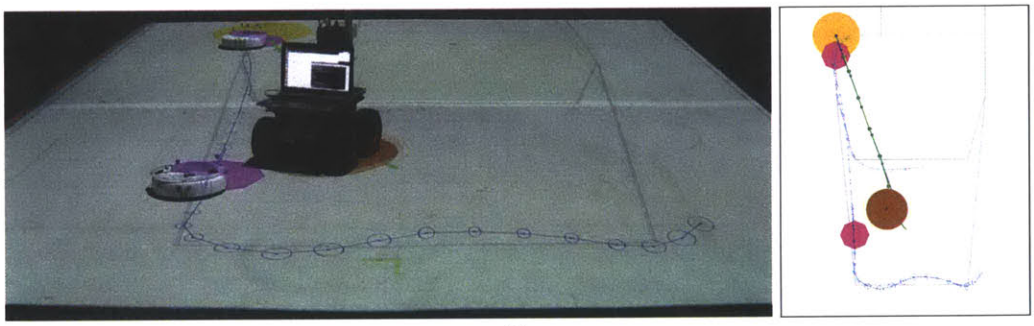
(c)



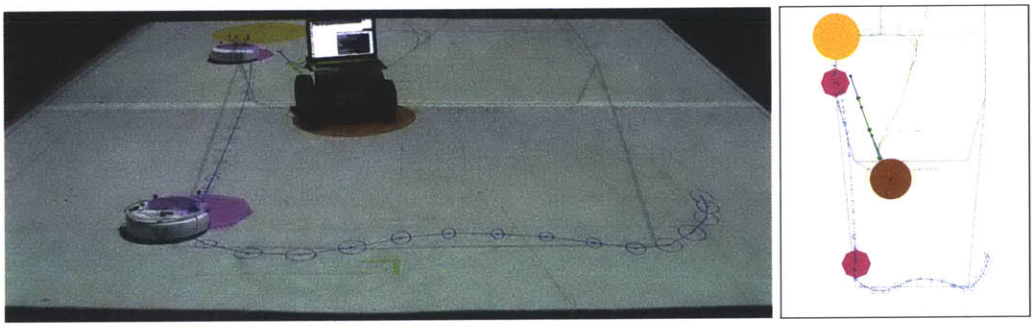
(d)



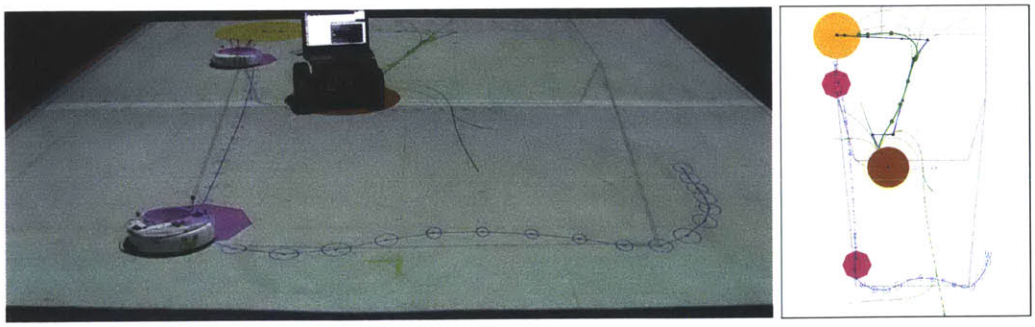
(e)



(f)



(g)



(h)

Figure 6-7: Moving rover planning paths around 2 dynamic robots

This scenario was executed for 2.5 minutes with no collisions.

Video 6 begins after Changepoint-DPGP has detected that the robot is executing a new behavior, as indicated by the velocity propagation predictions. After the robot has completed its first trajectory, the new behavior pattern is learned and incorporated in future predictions, (Figure 6-9a). This video is particularly interesting because the robot suffers from a control issue during the execution of its first trajectory, causing the learned behavior pattern to have a large kink (Figure 6-9b). Because the GP predictions smooth potentially noisy data, the predicted trajectory is smoothed as it becomes clear that the robot is traveling straight (Figure 6-9c). The predicted position given the portion of the trajectory in which the robot did not suffer from control issues is, as expected, a better representation of the actual trajectory executed by the robot (Figure 6-9d).

An additional scenario was executed, in which the robot similarly suffered from a control issue in the same location. In both scenarios, the robot alternated between the inner- and outermost behaviors, where the first used predictions from training data and the second was learned online. The average predictive error for these scenarios is presented in Figure 6-10. For both behaviors, the intent quickly converges to the correct behavior pattern after the vehicle has committed to one of the behaviors (Figure 6-10a). Notably, even though the known behavior starts with a higher prior probability (since more trajectories have been observed), the intent probability quickly drops to around 50% until the robot commits to a behavior. This is desirable, as the two behaviors initially overlap and are indistinguishable. The predictive error in the learned behavior is higher than that of the known behavior (Figure 6-10b) because the GP defining the learned behavior has fewer data points and observed trajectories suffered from several controller issues.

6.5 Summary

Experiments have been presented which motivate the need for predictive models of dynamic agents and evaluate Changepoint-DPGP on scenarios of varying complexity.

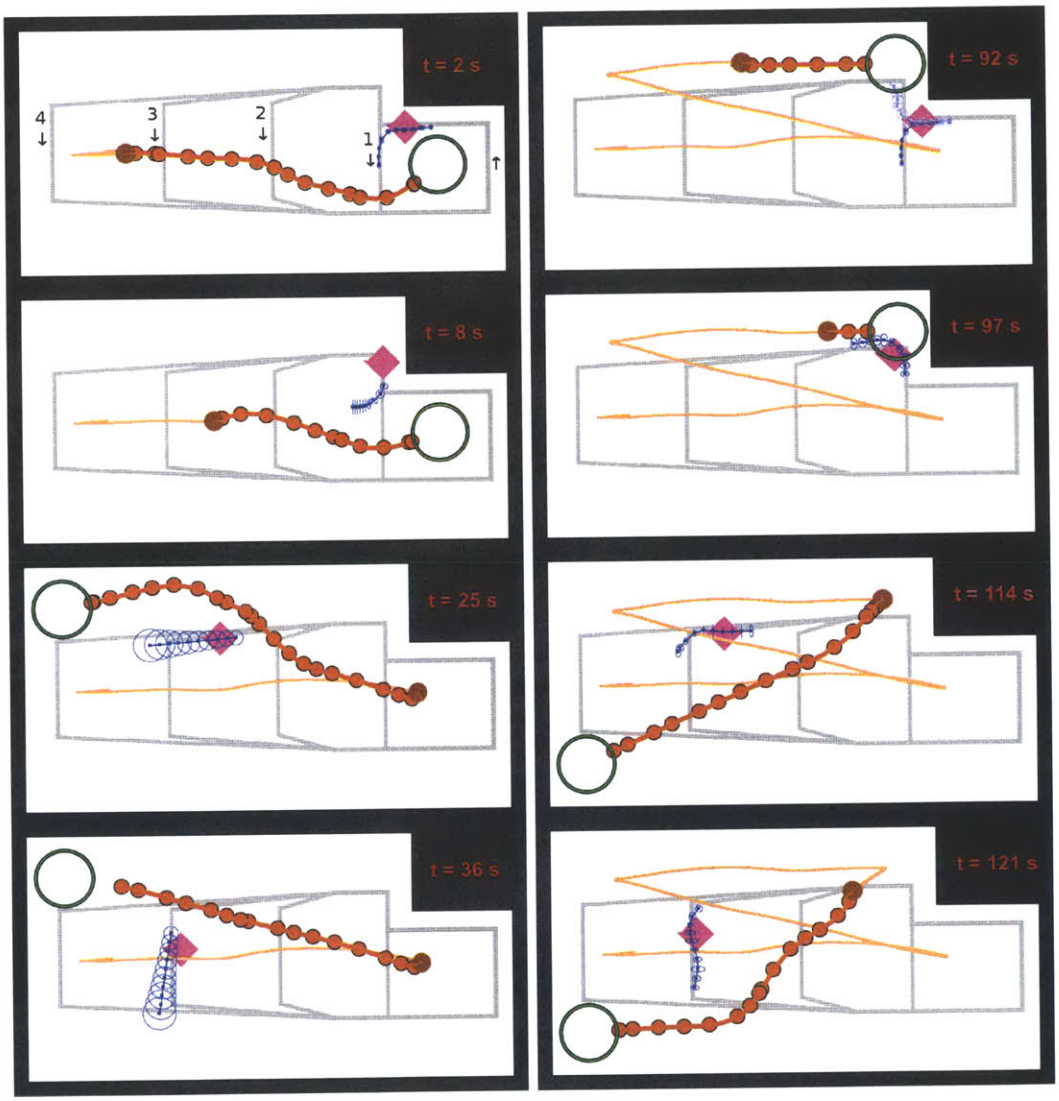
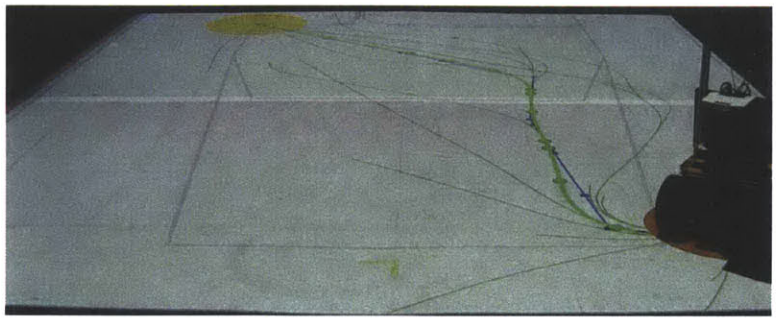
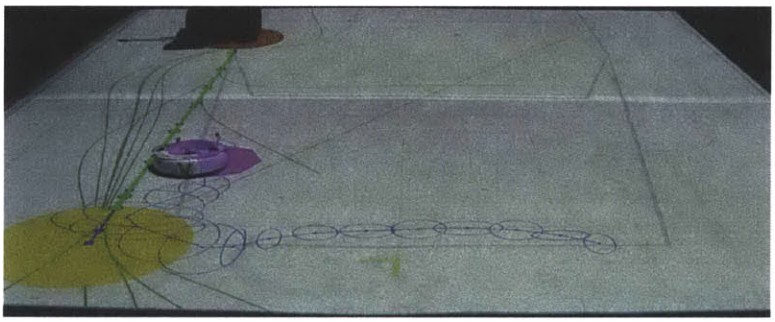


Figure 6-8: Illustrative example of Changepoint-DPGP executing online

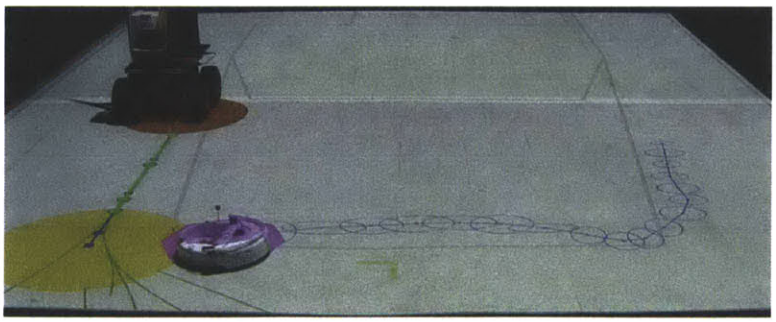


(a)

(b)

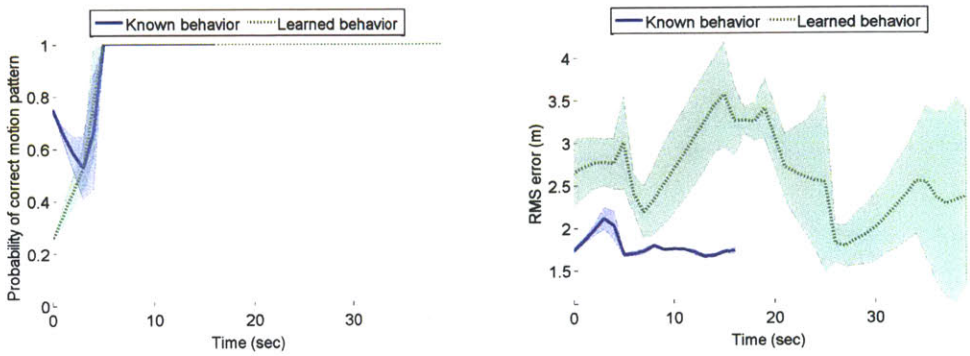


(c)



(d)

Figure 6-9: Online learning of new behavior



(a) Probability of correct motion pattern

(b) RMS predictive error

Figure 6-10: Prediction accuracy of known and learned behaviors

Results demonstrate that Changepoint-DPGP can be used to learn representative models of pedestrian and robot motion, from both offline and online data. Predictions generated from these models can be incorporated into a CC-RRT path planner to enable safe navigation through a dynamic world. The sampling-based nature of the motion planner allows for fast replanning in the presence of new and/or unexpected behaviors, such that a feasible path to a safe location within the environment is generated. This helps to prevent collision, as it is particularly unsafe to wait for updated path plans in an uncertain and dynamic world.

Chapter 7

Conclusions

This thesis has developed a framework for long-term trajectory prediction and robust avoidance of pedestrians and other dynamic agents in real-time, even when these agents exhibit previously unobserved behaviors and/or changes in intent. A key contribution is the Changepoint-DPGP algorithm, which uses a likelihood ratio test and offline clustering algorithm (DPGP) for efficient online classification of behaviors. This algorithm is able to learn new behavior patterns online and quickly detect and react to changes in intent. Unlike most approaches in the literature, accuracy in terms of both intent and trajectory prediction is considered; predictive results demonstrate improved accuracy over those approaches that do consider both forms of uncertainty.

These predictions are embedded within a chance-constrained motion planner (CC-RRT), such that probabilistically feasible trajectories can be identified in real time. Algorithms for pedestrian detection and autonomous navigation are implemented and developed. Experiments in several challenging environments demonstrate that this framework enables an autonomous rover to accurately predict the motion patterns of dynamic agents from various sources of sensor/perception data and safely navigate within a dynamic world.

7.1 Future Work

This section explores several ways in which the work in this thesis could be further extended.

7.1.1 Gaussian Process Predictions

In this thesis, GP predictions are tied to specific points within the environment, necessitating the collection of unique training trajectories for each new environment. A possible direction for future work is the generalization of the GP model to new environments. By relating training data to specific environmental features rather than a global coordinate frame, and relating features from one environment to another, the GP model from one environment can be scaled to fit another. This will make the predictions more applicable in the real world, such that data can be collected from a subset of environments and applied to a global scenario (e.g., training data from a subset of intersections within a city can be applied to generate GP models for all intersections).

7.1.2 Motion Planner Detection Uncertainty

As motivated by the pedestrian experiments in Section 6.2, in which detections were lost due to limited lidar field of view, an extension for the motion planner is also proposed. In addition to considering collision risk, a bound on belief distribution of obstacles within the environment can also be incorporated in the cost function. As the vehicle moves from node to node, the predictive distribution for the obstacles future position can be leveraged, such that the vehicle attempts to get updated measurements of particularly uncertain obstacles more frequently when these obstacles are outside of the sensor field of view or occluded. See [3, 12] for possible examples.

7.1.3 Efficient Detection and Tracking

Lastly, as discussed in Section 5.3.3, efficient detection and tracking of obstacles without the specification of shape, dynamics, and other heuristics remains an open problem. A possible direction is the use of efficient clustering methods (e.g., [13]) using Dependent Dirichlet Processes with the Generalized Polya Urn Dependent Dirichlet Process Mixture Model (GPUDDPM) algorithm [40].

Bibliography

[1] Motion Capture Systems from Vicon, 2011. 14 Minns Business Park, West Way, Oxford OX2 0JB, UK `http://www.vicon.com/`.

[2] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[3] Ali-akbar Agha-mohammadi, Suman Chakravorty, and Nancy Amato. FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *International Journal of Robotics Research (IJRR)*, 33(2):268–304, 2014.

[4] Omead Amidi and Chuck E Thorpe. Integrated mobile robot control. In *Fibers' 91, Boston, MA*, pages 504–523. International Society for Optics and Photonics, 1991.

[5] Georges S. Aoude, Brandon D. Luders, and Jonathan P. How. Sampling-based threat assessment algorithms for intersection collisions involving errant drivers. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, Lecce, Italy, September 2010.

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

[7] Max Bajracharya, Baback Moghaddam, Andrew Howard, Shane Brennan, and Larry H Matthies. A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle. *The International Journal of Robotics Research*, 28(11-12):1466–1485, 2009.

[8] Tirthankar Bandyopadhyay, Chong Zhuang Jie, David Hsu, Marcelo H Ang Jr, Daniela Rus, and Emilio Frazzoli. Intention-aware pedestrian avoidance.

[9] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-aware motion planning. In *Algorithmic Foundations of Robotics X*, pages 475–491. Springer, 2013.

[10] Michele Basseville and Igor V Nikiforov. Detection of abrupt changes: theory and applications. *Journal of the Royal Statistical Society-Series A Statistics in Society*, 158(1):185, 1995.

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

[12] B. Burns and O. Brock. Sampling-based motion planning with sensing uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3313–3318, Roma, Italy, April 2007.

[13] Trevor Campbell, Miao Liu, Brian Kulis, Jonathan P. How, and Lawrence Carin. Dynamic clustering via asymptotics of the dependent dirichlet process. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[14] Christophe Coué, Cédric Pradalier, Christian Laugier, Thierry Fraichard, and Pierre Bessière. Bayesian occupancy filtering for multitarget tracking: an automotive application. *The International Journal of Robotics Research*, 25(1):19–30, 2006.

[15] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[16] Marc P. Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In L. Bouttou and M. Littman, editors, *International Conference on Machine Learning (ICML)*, pages 225–232, Montreal, Canada, June 2009. Omnipress.

[17] Vishnu R. Desaraju. Decentralized path planning for multiple agents in complex environments using rapidly-exploring random trees. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, September 2010.

[18] David Ellis, Eric Sommerlade, and Ian Reid. Modelling pedestrian trajectory patterns with gaussian processes. In *Computer Vision Workshops (ICCV Work-*

shops), 2009 IEEE 12th International Conference on, pages 1229–1234. IEEE, 2009.

[19] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1056–1062, Nice, France, September 2008.

[20] Agathe Girard, Carl Edward Rasmussen, Joaquin Quintero-Candela, and Roderick Murray-smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, pages 529–536. MIT Press, 2003.

[21] Robert C. Grande. Computationally efficient Gaussian process changepoint detection and regression. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, June 2014.

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

[23] Peter Henry, Christian Vollmer, Brian Ferris, and Dieter Fox. Learning to navigate through crowded environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 981–986. IEEE, 2010.

[24] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2):51–64, April 2008.

[25] Tetsushi Ikeda, Yoshihiro Chigodo, Daniel Rea, Francesco Zanlungo, Masahiro Shiomi, and Takayuki Kanda. Modeling and prediction of pedestrian behavior based on the sub-goal concept. In *Robotics: Science and Systems*, 2012.

[26] Sony Electronics Inc. Sony vpl-fh31 projector, 2014.

[27] iRobot Corporation. iRobot Create programmable robot, 2010.

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

[29] Richard Kelley, Monica Nicolescu, Alireza Tavakkoli, C King, and G Bebis. Understanding human intentions via hidden markov models in autonomous mobile robots. In *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pages 367–374. IEEE, 2008.

[30] J. K. Kuchar and L. C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189, 2002.

[31] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. P. How. Motion planning in complex environments using closed-loop prediction. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Honolulu, HI, Aug 2008. (AIAA-2008-7166).

[32] R. Lachner. Collision avoidance as a differential game: Real-time approximation of optimal strategies using higher derivatives of the value function. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2308–2313, 1997.

[33] Velodyne Lidar. Hdl-32e lidar, 2012.

[34] Brandon Luders, Mangal Kothari, and Jonathan P. How. Chance constrained RRT for probabilistic robustness to environmental uncertainty. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Toronto, Canada, August 2010. (AIAA-2010-8160).

[35] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *Aerospace and Electronic Systems, IEEE Transactions on*, 34(1):103–123, 2002.

[36] Bernard Michini, Mark Cutler, and Jonathan P. How. Scalable reward learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.

[37] T. Miloh and S.D. Sharma. *Maritime collision avoidance as a differential game.* Institut fur Schiffbau der Universitat Hamburg, 1976.

[38] Adept MobileRobots. Pioneer 3-at, 2013.

[39] Luis E Navarro-Serment, Christoph Mertz, and Martial Hebert. Pedestrian detection and tracking using three-dimensional ladar data. *The International Journal of Robotics Research*, 29(12):1516–1528, 2010.

[40] Willie Neiswanger and Frank Wood. Unsupervised detection and tracking of arbitrary objects with dependent dirichlet process mixtures. *arXiv preprint arXiv:1210.3288*, 2012.

[41] S. Park, J. Deyst, and J. P. How. Performance and Lyapunov stability of a nonlinear path-following guidance method. *AIAA Journal on Guidance, Control, and Dynamics*, 30(6):1718–1728, November-December 2007.

[42] WD Penny and SJ Roberts. Bayesian multivariate autoregressive models with structured priors. Technical report, Oxford University, 2000.

[43] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[44] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, 2009.

[45] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, December 2005.

[46] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[47] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[48] SICK. Sick lms-291, 2014.

[49] H.W. Sorenson. *Kalman filtering: theory and application*. IEEE, 1985.

[50] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation.

[51] M. Valenti, B. Bethke, G. Fiore, J. P. How, and E. Feron. Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Keystone, CO, August 2006 (AIAA-2006-6200).

[52] Dizan Vasquez, Thierry Fraichard, and Christian Laugier. Incremental learning of statistical motion patterns with growing hidden markov models. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3):403–416, 2009.

[53] Yali Wang and Brahim Chaib-draa. A marginalized particle gaussian process regression. In *NIPS*, pages 1196–1204, 2012.

[54] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. Computational rationalization: The inverse equilibrium problem. *arXiv preprint arXiv:1308.3506*, 2013.

[55] Qiuming Zhu. Hidden markov model for dynamic obstacle avoidance of mobile robot navigation. *Robotics and Automation, IEEE Transactions on*, 7(3):390–397, 1991.

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