HUMAN MOTION BEHAVIOUR AWARE PLANNER
(HMBAP) FOR PATH PLANNING IN DYNAMIC
HUMAN ENVIRONMENTS

by

Siddharth Oli
B.Tech., Motilal Nehru National Institute of Technology, 2009

a THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

© Siddharth Oli 2014
SIMON FRASER UNIVERSITY
Spring 2014

All rights reserved.
However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
APPROVAL

Name: Siddharth Oli

Degree: Master of Applied Science

Title of Thesis: Human Motion Behaviour Aware Planner (HMBAP) for Path Planning in Dynamic Human Environments

Examining Committee: Dr. Shawn Stapleton, Professor  
Chair

Dr. Kamal Gupta, Senior Supervisor  
Professor, Engineering Science,  
Simon Fraser University

Dr. Greg Mori, Supervisor  
Associate Professor, Computing Science,  
Simon Fraser University

Dr. Carlo Menon, P. Eng., Internal Examiner  
Associate Professor, Engineering Science,  
Simon Fraser University

Date Approved: April 23, 2014
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files ("Work") (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU's own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU's rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author's written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author's knowledge, infringe upon anyone's copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013
Abstract

For a robot navigating in a human inhabited dynamic environment, the knowledge of how the robot’s movement can influence the trajectory of people around it can be very valuable. In this work we present a Human Motion Behaviour Aware Planner (HMBAP) which incorporates a Human Motion Behaviour Model (HMBM) in its planning stage to take advantage of this. HMBM is a potential field based obstacle avoidance model for people and the proposed planner uses it to give the robot a prediction of how people would react to its planned path. This information is useful for the robot to avoid imminent collisions with people in constricted spaces and the planner finds solutions in situations - called freezing robot problem - where past methods fail to find a solution. The resulting robot behaviour is also similar to how a human would move (in terms of avoidance behaviour) in a similar situation. We believe that this is a desirable feature for a robot navigating in a human inhabited environment. We have implemented HMBAP in simulation and also on the real robot in the RAMP Lab. Both simulations and experiments show the effectiveness of HMBAP.
Dedicated to my parents, grandma, Saumya and Divya.
Acknowledgments

I would like to thank Dr. Kamal Gupta, for his guidance and being a constant source of motivation; Bruno Blesphera, for the countless discussions on planners; Vinay Pilania, Gamini Siriwardana and Mike Hegedus, for advice on the thesis and their help with experiments; and finally my friends and family for their help and support.
# Contents

Approval ii  
Partial Copyright License iii  
Abstract iv  
Dedication v  
Acknowledgments vi  
Contents vii  
List of Tables ix  
List of Figures x  

## 1 Introduction 1  
1.1 Motivation ........................................ 1  
1.2 Background ....................................... 3  
1.3 Related work ..................................... 5  
1.4 HMBAP structure .................................. 7  
1.5 Thesis Contribution and organization .............. 8  

## 2 Human Motion Behaviour Aware Path Planner 10  
2.1 Human Motion Behaviour Model (HMBM) ................. 10  
2.2 Dynamic Window Approach (DWA) ........................ 13  
2.2.1 DWA for dynamic environments .................... 14
2.3 Navigation Function NF1 .................................................. 14
2.4 HMBAP Implementation ............................................... 15
2.5 Complexity Analysis ...................................................... 19

3 Simulations ................................................................. 22
  3.1 Simulation setup ......................................................... 22
  3.2 Simulation 1: Human waits for robot to move ..................... 22
  3.3 Simulation 2: Corridor passing situation ......................... 25
  3.4 Simulation 3: Crowded mall situation ............................ 26

4 Implementation ............................................................ 29
  4.1 Robotic Platform ....................................................... 29
  4.2 Detecting and tracking people ...................................... 29
  4.3 Localization and mapping ........................................... 35
  4.4 Planner implementation in ROS ................................. 37

5 Experiments ................................................................. 40
  5.1 Experimental setup .................................................. 40
  5.2 Experiment 1: Passing a human in a corridor .................. 41
  5.3 Experiment 2: Robot at a 4-way crossing ....................... 42

6 Conclusion and future work ........................................... 48
  6.0.1 Conclusion ......................................................... 48
  6.0.2 Future work ....................................................... 49

Appendix A ................................................................. 51

Bibliography ............................................................... 55
List of Tables

2.1 Variable Description ..................................................... 19

4.1 Best features for leg detection ........................................ 33
4.2 Example for set of assignments ........................................ 34

A.1 Parameters values for HMBAP ........................................... 53
A.2 Parameters for detecting humans using Kinect .................... 53
A.3 gmapping parameters ....................................................... 54
A.4 move_base parameters ...................................................... 55
A.5 leg_detector parameters ................................................... 55
## List of Figures

1.1 Effect of human following robot ........................................... 2  
1.2 Incorporating human model in planning ................................. 9  
1.3 HMBAP structure ................................................................ 9  

2.1 Human motion behaviour model demonstration ....................... 12  

3.1 Simulation 1 - Human waits for robot to move ....................... 24  
3.2 Simulation 2 - Corridor passing situation ............................. 26  
3.3 Simulation 3 - Crowded mall situation ................................. 28  

4.1 Robotic platform .................................................................. 30  
4.2 Object detection using Kinect ............................................. 32  
4.3 Map of Engineering Science Lab ......................................... 37  
4.4 Directed graph showing all nodes and the communication links between them. 39  

5.1 World maps showing robot’s initial position and goal ............ 41  
5.2 Experiment 1: Passing a human in a corridor - Without HMBM .... 43  
5.3 Experiment 1: Passing a human in a corridor - With HMBM ....... 45  
5.4 Experiment 1: Robot at crossing - Without HMBM ............... 46  
5.5 Experiment 1: Robot at crossing - With HMBM .................... 47
Chapter 1

Introduction

This chapter begins by explaining the motivation behind Human Motion Behaviour Aware Planner (HMBAP) and the need to consider robot’s trajectory influence on human motion behaviour while planning. We review existing literature on robot motion planning and discuss related work for path planning in dynamic human environments. Next we introduce HMBAP and discuss its overall structure. The chapter concludes by detailing the contribution of this work and the organisation of this work.

1.1 Motivation

As personal robots are becoming more popular, more and more robots are leaving the laboratories and entering consumer homes and office spaces. One of the most popular of these robots is the Roomba [1], an autonomous robotic vacuum cleaner from iRobot which has sold over 10 million units worldwide as of Feb 2014 [1]. The beam Pro [2] is a telepresence robot from Suitable Technologies, which is making it possible for people to remotely attend events across countries via the robot. One of the basic requirements of these robots is to be able to safely navigate human environments while avoiding collisions with humans moving around them. This problem was first encountered by RHINO [3] and MINERVA [4], the first robots to be deployed in dynamic human environments. Both robots acted as tour guide robot in museums and presented the challenges of navigating human crowds, inspiring extensive research in human motion behaviour and robot navigation in amongst moving humans.
CHAPTER 1. INTRODUCTION

The process of a human walking amongst other humans is highly interactive, and is based on a mutual understanding of behavioural aspect of human motion. For a robot to move and work among humans, it should have a certain level of understanding of human behaviour. Early methods for robot navigation in human environments depended on **proxemics**, a study of non-verbal communication through touch or body movement first introduced by E.T. Hall [5]. Recent methods emphasize on accurately predicting human motion, and then using the predicted motion, for planning [11]. While these methods do incorporate human motion information in planning, they do not consider that the robot’s planned path may actually affect the trajectories of the humans around the robot.

The advantage of considering robot’s planned path’s influence on human motion can be explained by the navigation problem shown in Fig. 1.1. The robot is shown here in yellow exiting a hallway. Since the robot does not have vision of the occluded areas near the exit of the hallway the best option for the robot is to slow down and exit the hallway, minimising its chances of colliding with humans walking in the occluded regions. Now consider the case in Fig. 1.2 (b), in this case the robot is followed by a human. The robot now sees this human walking towards him and in order to avoid a collision, it exits the hallway risking a collision with occluded moving humans in the hallway. The ideal solution in this case would have been for the robot slow down and expect the person behind him to slow down too and safely exit the hallway. This behaviour is possible when the robot has the understanding
that its actions can influence human motion behaviour.

We are interested in incorporating the influence of the planned motion of the robot on the humans around it, in the path planning process for the robot. For this, we use a Human Motion Behaviour Model (HMBM) (based on a model that was first proposed in [14]) which predicts a person’s motion based on the motion of both the robot and other humans around the subject. We integrate this model in our planning stages. The block diagram in Fig. 1.2 shows the difference in the incorporation of a human motion model in our planner and earlier planners, it can be seen that our planner uses the human behaviour model as a sub-module rather than as one time input. The result of this deeper integration is that the robot, while planning a path, considers how its planned path would influence the behaviour of the humans around it. This information is used to evaluate and select the robot’s planned path, thereby resulting in more informed robot motions.

The integration of HMBM with the planner also solves the Freezing Robot Problem (FRP) as noted, among others, by Trautman and Krause [13]. In dense crowds of people, due to high obstacle density, and motion uncertainty of humans, conventional path planning approaches often fail to find a collision free path. Thus, the robot either makes no forward progress or takes extreme evasive action to avoid collisions. The reason for this is that conventional algorithms consider humans to be blind dynamic obstacles; that is, they do not consider that the robot’s motion may actually influence a person’s behaviour. By using HMBM in the planning stages, HMBAP gives the robot the reasoning that people try to avoid collisions with the robot. This new knowledge opens new paths for the robot which were considered not collision free without the use of HMBM. This helps the robot handle tough situations which conventional methods find hard to navigate.

1.2 Background

In motion planning, the notion of a configuration is the minimum number of parameters needed to completely specify a robot’s position and orientation in space. For a mobile robot, it corresponds to the pose of the robot (x,y,θ). The basic motion planning problem is to produce a continuous motion that connects a start configuration of the robot to a given goal configuration, while avoiding collision with known obstacles. The motion of the robot is represented as a path in configuration space, the space of all possible configurations of a robot where a configuration describes the pose of the robot. The literature on robot motion
planning [6] [7] can be broadly divided into three main sections - geometric algorithms, potential based methods, and sampling based methods. Low-dimensional problems can be solved with grid-based algorithms or geometric algorithms, whereas, sampling based methods are most suited to high-dimensional problems. Here we briefly describe the key approaches that we have used in our work.

One approach to solving the basic motion planning problem is to treat the robot’s configuration as a point in a potential field that simulates attraction to the goal (a valley), and repulsion from obstacles (hills). The robot moves under the influence of this potential field toward the minimum (ideally global) and the resulting trajectory is output as the path. This approach has advantages in that it is computationally fast. However, the robot can become trapped in local minima of the potential field, and fail to find a path. It is therefore prudent to define potential functions that have no local minima, such functions are called navigation functions. Navigation function I (NF1) [22] is one such function which can be constructed for a discretized representation of the configuration space (hence is practical for low dimensional C-space only). Starting from the goal position, each grid cell is marked with its $L^1$ distance from the goal position, we progress to the neighboring cells of the goal cell and to the subsequent cells in the form of a wave-front expansion; we stop the expansion at cells occupied by obstacles. The result is a local minima free potential function with its absolute minimum at the goal position. Once the navigation function is computed, the global path is obtained by running a best first search (on $L^1$ distance) on the grid cells from the start to the goal.

Dynamic window approach (DWA) [21] is a velocity space based obstacle avoidance technique which incorporates the dynamics of the robot by restricting the velocity search space to the set of reachable velocities under the dynamic constraints. The velocity search space is further limited to admissible velocities. A velocity is admissible if the robot following the trajectory described by the velocity can stop before hitting an obstacle. Each velocity in this velocity search space represents a possible trajectory of the robot which is obtained assuming piece-wise constant velocities. Among these possible trajectories we then search for the trajectory with minimum cost. The cost function is defined over the trajectories to optimize motion planning goals like distance from goal G and distance from obstacles. Selecting the minimum cost trajectory at every time step gives us the overall trajectory of the robot from start to goal as sequence of circular and/or straight line sub-trajectories.

DWA is a reactive avoidance technique, and it can deal with dynamic obstacles in the
workspace by relying on its fast on-line replanning. However, in highly dynamic environment it becomes important to incorporate the dynamics of moving obstacles in the motion planner. An extension of DWA, incorporating moving obstacles, is demonstrated by Seder et al [23]. In this extension dynamic obstacles are regarded as occupied moving cells in a grid map. The predicted trajectory of each moving cell is used for the cost calculation for the possible robot trajectories set generated by DWA. This ensures the trajectories are evaluated against the future time varying position of obstacles rather than considering them as static. This method is seen to produce smoother and shorter paths for the robot, in dynamic environments. Dynamic Window Approach and Navigation Function are the two core planners that we use in our overall HMBAP approach.

1.3 Related work

Early work in the field of robot navigation in human populated environment has been on building social rules a robot is expected to follow amongst humans. Pacchierotti et al [10] explain the notion of intimate, personal, social and public spaces with respect to a human subject. Their work studies the problem of passing a human in a hallway and propose a control strategy based on the rules of proxemics [5]. While the proposed control strategy works well to generate human-like behaviour for the robot, the method limited to hallway passing situations. Also the method cannot be generalised to work in presence of multiple human subjects. Our planner addresses similar hallway passing situations but it is not limited to hallway passing and considers multiple humans walking in it’s vicinity.

A social cost map based approach is proposed in [15]. It defines a safety criterion and visibility criterion for creating a social cost map for robot navigation. While the safety criterion attaches a high cost on positions close to humans, the visibility criterion attaches a high cost to regions which are not in the field of view of the humans. This method results in robot behaviours such that the robot does not approach people from behind or come to close to them. The authors further extend their method to dynamic scenarios and show similar utility of the method. The method however considers humans as static while planning, and depends on fast on-line re-planning to account for changes in their position. For this reason this method is not suitable safe navigation in dynamic environments as explained in [12]. The idea of a safety and visibility criterion however, are important for navigation in dynamic human environments and can be developed further.
Human motion behaviour could be learned from real world data. Bennewitz et al [11] use a Hidden Markov Model (HMM) to estimate the current and future positions of persons based on sensory input. While this method gives a good estimation of pedestrian trajectories, it does not consider the robot’s influence on the pedestrian trajectories. Luber et al [16] use real world data to study pairwise relationships between motion of people. They develop an unsupervised learning approach to learn human motion patterns by considering relative angle of approach between two passing humans. The method is shown to give better results than a *proxemics* based method but the method is limited to two human subjects. Also it will be interesting to see how this method generalizes to workspaces different from the one it is trained in.

The importance of considering the environment objects’ future behaviour has been pointed out by Fraichard [12]. Fraichard reasons that majority of the current robotic systems work in human environments only because *people themselves took care of the collision avoidance*. Had these robotic systems been placed among blind people collision would have happened. It is clear here, that the robot’s trajectory affects the human trajectories, and since that it the case, the robot should account for this characteristic in its planning. By using HMBM in our planner we incorporate this collision avoidance behaviour of people at the planning stage for better evaluation of robot’s planned paths.

Methods based on *Velocity Obstacle (VO)* [17], [18], have been used for multi-agent path planning in dynamic environment. These methods show that joint collision avoidance results in improved efficiency at joint navigation tasks in multi-robot systems. *Reciprocal Obstacle Velocity* [17] approach is guaranteed to be collision and oscillation free (oscillations result when two robots attempt to pass one another in the same direction). In case of robot navigation in human crowds this approach could be used, but this method makes the assumption that all agents choose velocities in a pre-specified manner which cannot be said for humans. We cannot assume that humans will make the same collision avoidance maneuver as the robot at every planning step.

Trautman and Krause [13] apply the idea of joint collision avoidance for navigation in dense crowds. This work is most relevant to our approach as it also considers the influence of the robot’s planned trajectory on the future motion of humans. However, the approach to incorporate this influence while planning is very different. Trautman et al represent observed and future object trajectories as Interacting Gaussian Processes (IGPs) with the goal positions incorporated as future sample points in the trajectories. The idea of IGPs
is such that individual human trajectories interact with each other such as to maximise distance between two human positions at any time. In this representation the robot’s trajectory is considered no different than a human trajectory and similar method is used to estimate both the trajectories. By using this representation the problem of path planning for the robot is solved as a trajectory prediction task for all agents, using the IGP model. This approach is specifically designed to solve the problem of robot navigation in open environment, with no static obstacles, among human crowds. The authors train and test their method in a cafeteria scenario by using overhead cameras and learning crowd behaviour in the cafeteria. Our method on the other hand is a more planner oriented, model based approach which uses HMBM for predicting people behaviour in different navigation scenarios and uses a dynamic window approach for planning, hence accounting for the dynamics of the robot. We use on-board sensors for sensing and plan for static obstacles/walls in our planner. The planner also explicitly account for the fact that humans can be unaware of the robot’s presence and might not always avoid the robot.

### 1.4 HMBAP structure

Fig 1.3 describes the structure of HMBAP. A static world map of the environment is build using the robot sensors. This map is fed to a Navigation Function based planning method to generate a path from start to goal. This path is fed to the local planner which moves the robot along this path while avoiding collisions with human subjects in the environment. Dynamic window approach is used to generate possible trajectories for the robot and for each of these trajectories HMBAP is used to generate future trajectories for humans around the robot. Costs for each of the possible robot trajectories are evaluated considering the corresponding generated human trajectories and the trajectory with minimum cost is selected for execution.

The human behaviour model we use for our planner is a potential field method based on social forces. This model is derived from the one introduced in [14]. It depicts the fact that people move in the general direction of their goal, while trying to avoid collisions with obstacles. It defines potentials which on one hand, help it maintain its constant velocity, and on the other, repel it from obstacles. For our purpose, one of the important interacting forces in the model is the repulsive force exerted on the human subject by the robot. It is this force which helps the robot estimate its influence on objects and plan accordingly.
CHAPTER 1. INTRODUCTION

One major advantage of this model is that it uses only the current positions/velocities of the objects to predict their next state. This results in the fast computation of the model at every step, and it is efficiently integrated in the planning routine.

It is important to consider that not all humans may follow HMBM. Some people are distracted or simply unaware of presence of other moving objects. We generalize our approach by classifying people as aware or unaware of the robots presence. While a human who is aware of the robot would try to avoid a collision with it, a human who is unaware would not do so. The classification whether a human is aware (or unaware) of the robot could potentially be made based on the gaze direction of the person [19], and by tracking its trajectory [20]. This is an open research area and is outside the scope of this thesis. In our work the information of humans being aware or unaware is input to the planner as predetermined flags. This classification produces interesting robot behaviour and results in safer trajectories. Chapter 2 explains these modules in detail.

1.5 Thesis Contribution and organization

The key contribution of our work is a novel planner (HMBAP) for a robot in a dynamic human environments, that considers the influence of the robot’s planned trajectory on nearby humans. Our planner uses a Human Motion Behaviour Model (HMBM) in our planner to achieve this. The method is reactive, as HMBM is fast to compute, and incorporates the distinction that humans can be aware or unaware of the robot’s presence. Initial results of this work have been published in [26].

The structure of the remainder of this thesis is as follows. Chapter 2 explains our approach in detail. It explains the Human Motion Behaviour Model (HMBM), the reactive local planner, the global planner, and finally the algorithm for the complete planner. Chapter 3 discusses test results of simulated situations and compares the performance of our planner with and without HMBM. Chapter 4 explains the implementation of the method on our lab’s laptop. Chapter 5 presents results from experiments and we conclude in Chapter 6 by summarising and discussing future work.
CHAPTER 1. INTRODUCTION

(a) Human Motion Behaviour Aware Path Planner

(b) Previous planners

Figure 1.2: Incorporating human model in planning

Figure 1.3: HMBAP structure
Chapter 2

Human Motion Behaviour Aware Path Planner

This section explains in detail the modules in HMBAP. We first explain the Human Motion Behaviour Model (HMBM), followed by DWA, its extension to moving obstacles, Navigation Function NF1 and the final integration of HMBM with DWA and NF1.

2.1 Human Motion Behaviour Model (HMBM)

The motion behaviour for people is dependent on various social/environmental factors. While it is tough to model human behaviour as a whole, some universal basic traits can be modelled using a social force model, proposed in [14]. The model considers human motion as holonomic. According to this model, the changes $d\vec{v}_h/dt$ of the preferred velocity $\vec{v}_h(t)$ of a human $h$ are described by a vector $\vec{F}_h(t)$ that can be interpreted as social force. $\vec{F}_h(t)$ describes the motivation of the pedestrian to react to perceived information. We consider three main characteristics of human behaviour:

i) Unobstructed movement - If a person’s motion is not disturbed, he/she will walk in a desired direction with a desired speed $\vec{v}_h^0(t)$. A deviation of the actual velocity $\vec{v}_h(t)$ from the desired velocity $\vec{v}_h^0(t)$, due to necessary obstacle avoidance measures, leads to a tendency to approach $\vec{v}_h^0(t)$ again within a certain relaxation time $\tau_h$. This can be described by an
CHAPTER 2. HUMAN MOTION BEHAVIOUR AWARE PATH PLANNER

acceleration term of the form:

\[ \vec{F}_0^h(\vec{\vartheta}_h(t), \vec{\vartheta}_0^h(t)) = \frac{1}{\tau_h} \cdot (\vec{\vartheta}_0^h(t) - \vec{\vartheta}_h(t)) \quad (2.1) \]

According to this model humans after performing a obstacle avoidance maneuver will continue to move in a direction parallel to their original direction of motion. This will be the case when the goal of the subject is a long way off. Since exact goal information for humans is mostly not available we approximate unobstructed human motion by a preferred velocity direction.

ii) Influence of other moving obstacles - The motion of person is influenced by other moving obstacles (humans/robots). A person \( h \) moves in such a way so as to avoid getting close to other moving objects. We model this tendency in humans by defining a repulsive force acting on human \( h \) due the presence of obstacle \( k \). This repulsive force is described by the acceleration term:

\[ \vec{F}_{rep}^{h,k}(\vec{r}_{h,k}) = w_{rep} \cdot \left( \frac{1}{|\vec{r}_{h,k}| - d_{col}^{h,k} - d_{thresh}} \right) \cdot (\vec{r}_{h,k}) \quad (2.2) \]

Where, \( \vec{r}_{h,k} \) is the position vector of \( k \) with respect to \( h \). \( d_{col}^{h,k} \) is the collision distance between the two objects; this distance is equal to radius of \( h \) + radius of \( k \), under the assumption of circular geometry for subjects. \( d_{thresh} \) is the threshold distance; we consider repulsive forces of \( k \) acting on \( h \) only when the distance between \( k \) and \( h \) is less than \( d_{thresh} \). \( w_{rep} \) is a weight which is used to adjust the magnitude of the repulsive force.

iii) Influence of static obstacles - Another source of social forces is the static obstacles in the workspace, for example, walls in indoor environments. HMBM deals with walls by setting a person’s velocity component normal to the wall to zero, when he/she comes close to the wall. This ensures that people, when close to a wall, can only move parallel to the wall. The model can be extended to arbitrary static obstacles by adding a repulsive force acting on people, that is normal to the boundary of the static obstacles.

The net social force on person \( h \) then becomes:

\[ \frac{d\vec{\vartheta}_h}{dt} = \vec{F}_h(t) = \vec{F}_0^h(\vec{\vartheta}_h(t), \vec{\vartheta}_0^h(t)) + \sum_k \vec{F}_{rep}^{h,k}(\vec{r}_{h,k}) \quad (2.3) \]

Fig. 2.1 shows HMBM in action. The figure shows the robot (in yellow) and a human (in red) at current time \( t_c \). The blue dotted line is a possible trajectory for the robot for time
Figure 2.1: Human motion behaviour model demonstration

$t \in [t_c, t_c + t_{sim}]$, with each successive dot corresponding to the future robot position at time incremented by a certain fixed resolution of $\Delta t$. For each blue trajectory point at time $t$ we use the HMBM to obtain a prediction of the person’s location at time $t$. The green dotted line shows this predicted trajectory obtained corresponding to the robot’s blue trajectory. To show the working of the model, the figure shows the force vectors, $\vec{F}_h^0$ and $\vec{F}_{h,k}^{rep}$, at an arbitrary point on the green path, the path predicted by HMBM if the robot were to move along the blue trajectory. At each step HMBM considers these forces to determine the next position of the human obstacle.

**Not all humans follow HB model**

When humans navigate, they are constantly observing people around them to see if the person in their vicinity is aware of their presence or is distracted. A person who is distracted (unaware of the presence of other people) is more likely to bump into others, as he himself/herself does not try to avoid collisions. The robot needs to have a similar capability to make such discrimination since this information can be very important for safe navigation in crowded human environments.

To introduce this additional planning capability in our planner, we associate each human
obstacle in the robot’s sensor’s field of view with a *awareness* flag, which indicates whether it is *aware* or *unaware* of the robot’s presence. The criteria for setting these flags could be based on visual clues, for example, by looking at the person’s gaze direction [19]; and could also be inferred from a person’s trajectory [20]. For instance, if a person does not tend to slow down or move away while approaching the robot, its flag can be set to *unaware*. This is an open research area and is outside the scope of this thesis.

For this work we assume that the flag status is available and is provided to the robot. While a pedestrian, with its *awareness* flag set to *aware*, follows HMBM and is expected to avoid a collision with the robot; a pedestrian with its awareness flag set to *unaware* does not follow HMBM and does not try to avoid collisions. Interesting robot navigation behaviours are seen by using this discrimination, as is seen in simulations in Section 3.

### 2.2 Dynamic Window Approach (DWA)

DWA [21] is a velocity space based obstacle avoidance technique which incorporates the dynamics of the robot by restricting the velocity search space to the set of reachable velocities under the dynamic constraints. Let $T$ be the time interval during which accelerations $\dot{\vartheta}$ and $\dot{\omega}$ will be applied on the robot and let $v_c = (\vartheta_c, \omega_c)$ be the current robot velocity. The velocity space $V_d(\vartheta, \omega)$ is defined as follows:

$$V_d = (\vartheta, \omega) | \vartheta \in [\vartheta_c - \dot{\vartheta} * T, \vartheta_c - \dot{\vartheta} * T] \land \omega \in [\omega_c - \dot{\omega} * T, \omega_c - \dot{\omega} * T]$$ (2.4)

The trajectory of the robot is approximated by a sequence of circular and straight line assuming piece-wise constant velocities. Here the size of the search space is limited by $\dot{\vartheta} < \dot{\vartheta}_{\text{max}}, \dot{\omega} < \dot{\omega}_{\text{max}}, \vartheta < \vartheta_{\text{max}}$ and $\omega < \omega_{\text{max}}$ where $\dot{\vartheta}_{\text{max}}, \dot{\omega}_{\text{max}}, \vartheta_{\text{max}}$ and $\omega_{\text{max}}$ are dynamic constraints on the robot (see Table A.1). The velocity search space is further limited to admissible velocities $V_a$. A velocity is admissible if the robot following the trajectory described by the velocity can stop before hitting an obstacle. The resulting reachable velocity space is then $V_r = V_a \subseteq V_d$. An objective function which incorporates a criteria for choosing the optimal velocity is then defined over $V_r$. The objective function we use in our work is a cost function defined in Eq. 2.5. Optimizing the objective function over $V_r$ gives the optimal velocity.
2.2.1 DWA for dynamic environments

DWA is a reactive avoidance technique, and it can deal with dynamic obstacles in the workspace by relying on its fast on-line replanning. However, in highly dynamic environment it becomes important to incorporate the dynamics of moving obstacles in the motion planner. An extension of DWA, incorporating moving obstacles, as demonstrated by Seder et al [23], is seen to produce smoother and shorter paths for the robot, in dynamic environments. We use a similar approach to incorporate moving obstacles’ dynamics in our method. While Seder et al use a constant velocity model to estimate obstacle positions, we use HMBM to estimate human positions.

Our method modifies DWA by changing process of evaluating the objective function for each robot trajectory. A trajectory corresponding to an velocity \( v_r(\dot{\theta}, \omega) \) in the velocity space \( V_r \), is sampled at a time interval of \( \Delta t \) for a trajectory duration of \( t \in [t_c, t_c + t_{sim}] \). Let \( S_h(t) \) be the set of positions of all humans in the environment at time \( t \). \( S_h(t_c) \) then represents human positions at current time \( t_c \), i.e. the sensed human positions. Original DWA assumes that humans are static for \( t_{sim} \) time, and therefore uses \( S_h(t_c) \) for all \( t \in [t_c, t_c + t_{sim}] \) to evaluate the objective function for the trajectory. We modify this step to incorporate the dynamics of humans as follows. For each sampled point at time \( t \in [t_c, t_c + t_{sim}] \), the set of future human positions, \( S_h(t) \), are predicted using a dynamic model for humans. We use HMBM to obtain the predicted positions \( S_h(t) \). Now instead of using the sensed human positions, \( S_h(t_c) \), the predicted human positions \( S_h(t) \) are used for evaluating the objective function. This process ensures that the planner considers the dynamics of the moving obstacles while choosing a best trajectory.

2.3 Navigation Function NF1

DWA is susceptible to local minima with respect static obstacles. This limitation can be overcome by using a global planner along with DWA. A grid-based navigation function, NF1 can be used with DWA to achieve a global DWA [24]. [24] incorporates NF1 in DWA by adding a NF1 cost to the DWA cost function, we on the other hand use a more general approach. First NF1 plans a global path from the start to the goal, and the planned path is then piece-wise fed to the DWA planner which is used in the cost function of DWA. In this way NF1 can be easily be decoupled from DWA and replaced by any other path planner as per need.
The navigation function is computed with a wave front algorithm. Starting from the goal position, each grid cell is marked with its $L_1$ distance from the goal position. The result is a local minima free potential function with its absolute minimum at the goal position. Once the navigation function is computed, the global path is obtained by running a best first search (on $L_1$ distance) on the grid cells from the start to the goal. This planned path is passed to DWA.

### 2.4 HMBAP Implementation

The complete planner implementation is presented in Algorithm 1. When the robot is set a goal, the navigation function (NF1) plans a path from the current (start) pose to the goal pose. This planned path is computed with respect to the static obstacle map available with the planner and is used to avoid getting stuck local minima. At any current time $t_c$ the planner computes the velocity space $V_r$ consisting of admissible velocities in the dynamic window space. For each of the velocities $\nu_i \in V_r$, the planner simulates future trajectories $Tr$ for the robot by assuming constant velocity for $t_{sim}$ seconds. Each $Tr_i$ describes a circular arc or a straight line trajectory for duration $[t_c, t_c + t_{sim}]$ with a time step of $\Delta t$. For each of these trajectories, HMBM has a prediction of human trajectories. Using the corresponding predicted human trajectories, the planner evaluate the objective function, $TrCost_i$, for each trajectory $Tr_i$ corresponding to $\nu_i \in V_r$ and minimize it over $V_r$. $TrCost$ is defined as follows:

$$TrCost(\vartheta, \omega) = \alpha \cdot d_{goal}(\vartheta, \omega) + \beta \cdot d_{path}(\vartheta, \omega) + \gamma \cdot e^{-\eta[d_{obs}(\vartheta, \omega)-d_{col}]}$$  \hspace{1cm} (2.5)$$

Here $\alpha$, $\beta$ and $\gamma$ are constants used for tuning the cost function. The initial values of these constants were set as in the implementation of dynamic window approach for static scenarios in [25]. From there the values were tuned empirically via the simulations to obtain the desired robot behaviour seen in simulations in section 3 and experiments in section 5. $d_{goal}$, $d_{path}$ and $d_{obs}$ are described below:

1) Distance from goal ($d_{goal}$) is the distance of the last point in the trajectory $Tr_i$ from the goal position. This parameter measures how close the trajectory takes the robot to its goal. A lower $d_{goal}$ ensures that the trajectory gets the robot closer to the goal.

2) Deviation from global path ($d_{path}$) is the distance of the last point in the trajectory $Tr_i$ from the global path generated by the global planner. A lower $d_{path}$ ensures that the
selected trajectory deviates less from the generated global path, lest the robot gets trapped in a local minima.
Algorithm 1 Planner implementation

// $S_h(t), V_h(t)$ are sets of positions and velocities of all humans at time $t$
$\text{GPath} \leftarrow \text{NF1}(\text{StaticMap, Start, Goal})$
$s_r(t_c) \leftarrow \text{Start}$

while $s_r(t_c) \neq \text{Goal}$ do

  // Update pose/vel. for the robot and humans
  $[s_r(t_c), v_r(t_c)] \leftarrow \text{UpdateRobotOdometry}()$
  $[S_h(t_c), V_h(t_c)] \leftarrow \text{HumanTracking}()$
  $V_r \leftarrow \text{list of dynamically admissible velocities}$

  for all $\nu_i \in V_r$ do

    for $t = t_c \rightarrow t_c + t_{\text{sim}} \text{ step } \Delta t$ do

      if $\text{Collision}(s_r(t), \text{StaticMap}, S_h(t)) = \text{true}$ then

        collision = true

        break

      else

        $d_{\text{obs}} = \min[d_{\text{obs}}, d_{\text{obs}}(t)]$
        $\text{Tr}_i \leftarrow \text{push}(s_r(t_c))$ // $\text{Tr}_i$ is the traj. w.r.t. to $\nu_i$
        $[S_h(t + \Delta t), V_h(t + \Delta t)] = \text{HMBM}(S_h(t), V_h(t), s_r(t), \text{StaticMap})$
        $s_r(t + \Delta t) \leftarrow \text{UpdateRobotPosition}(\nu_i)$

      end if

    end for

    if collision = true then

      $\text{TrCost}_i = -1$

    else

      evaluate $\text{TrCost}_i$ as in Eq 2.5

    end if

    $\text{TrSet} \leftarrow \text{push}(\text{Tr}_i, \text{TrCost}_i)$

  end for

  $\text{BestTr} = \text{MinimumCost}(\text{TrSet})$
  $\text{Execute}(\text{BestTr})$

end while

// $\text{Collision}(s_r(t), \text{StaticMap}, S_h(t))$ checks if robot is in collision by using assuming a circular robot geometry
3) Minimum distance to obstacles ($d_{\text{obs}}$) is the minimum distance to obstacles along the entire trajectory. The exponential function in Eq. 2.5 represents the cost due to this distance. The cost function is inversely proportional to $d_{\text{obs}}$. A larger $d_{\text{obs}}$ results in lower cost ensuring the robot avoids obstacles by a greater/safer distance. Here $d_{\text{col}}$ ($d_{\text{col}} < d_{\text{obs}}$) is the collision distance for the robot and $\eta$ is a constant (see Table A.1).

We use HMBM while evaluating the objective function $TrCost_i$ for a trajectory $Tr_i$. In the constituent terms of $TrCost_i$, $d_{\text{goal}}$ and $d_{\text{path}}$ are independent of HMBM, as they only depend on the position of the robot at the completion of trajectory $Tr_i$. $d_{\text{obs}}$ however, which measures how close the trajectory gets to obstacles, makes use of HMBM to estimate predicted human locations and their distance from the robot for every time step $t \in [t_c, t_c + t_{\text{sim}}]$. The process starts with the set of positions $S_h(t_c)$ and the velocities $V_h(t_c)$ of humans at $t_c$, available through sensors/simulator. For obstacle positions $S_h(t_c)$ and robot position $s_r(t_c)$, $d_{\text{obs}}(t_c)$ is evaluated. For the next time step the new obstacle positions and velocities, $S_h(t_c + \Delta t)$ and $V_h(t_c + \Delta t)$, are determined using HMBM.

$$[S_h(t + \Delta t), V_h(t + \Delta t)] = HMBM(S_h(t), V_h(t), s_r(t))$$ (2.6)

Implementation of HMBM in given in Algorithm 2. Thus obtained $S_h(t_c + \Delta t)$ and $V_h(t_c + \Delta t)$, and $s_r(t_c + \Delta t)$ for $Tr_i$, are used to evaluate $d_{\text{obs}}(t_c + \Delta t)$. This process is iterated for all the points in the interval $[t_c, t_c + t_{\text{sim}}]$. The minimum value of $d_{\text{obs}}(t)$ is set to be the value of $d_{\text{obs}}^{\text{min}}$ for trajectory $Tr_i$. The values of $d_{\text{goal}}$ and $d_{\text{path}}$ are evaluated after $d_{\text{obs}}^{\text{min}}$ is obtained. Together the three quantities give the $TrCost_i$ of trajectory $Tr_i$. Once the cost of all trajectories corresponding $\nu_r \in V_r$ are obtained, the cost function is minimized over $V_r$ to get the optimal velocity. This process is repeated at each time step, generating optimal velocity for execution, which is passed to the robot’s motor control module.
Algorithm 2 HMBM implementation

\[ [S_h(t + \Delta t), V_h(t + \Delta t)] = HMBM(S_h(t), V_h(t), s_r(t), flags(t)) \]

// \( S_h(t), V_h(t) \) are the set of positions and velocities of human \( h \) at time \( t \)

// \( flags(t) \) is the set of awareness flag values at time \( t \)

for \( h \) in all moving obstacles

\[ \ddot{a}c_h^0 = (\ddot{v}_h(t) - \ddot{v}_h(t))/\tau \]

\[ \ddot{a}c_h^{rep} = 0 \]

for \( k \) in all moving obstacles \( \neq h \)

\[ \ddot{a}c_{h,k}^{rep} = w^{rep} \cdot \left( 1/(||\vec{r}_{h,k}|| - d_{col}^{h,k}) - 1/(d_{thresh}^{h,k}) \right) \cdot (\ddot{r}_{h,k}) \]

\[ \ddot{a}c_h^{net} = \ddot{a}c_h^0 - \ddot{a}c_h^{rep} \]

if \( flag_h = 0 \) then \( \ddot{a}c_h^{net} = 0 \)

\[ \ddot{v}_h(t + \Delta t) = \ddot{v}_h(t) + \ddot{a}c_h^{net} \cdot \Delta t \]

\[ \ddot{s}_h(t + \Delta t) = \ddot{s}_h(t) + \ddot{v}_h(t) \cdot \Delta t + 1/2 \cdot \ddot{a}c_h^{net} \cdot \Delta t^2 \]

\[ S_h(t + \Delta t) \leftarrow push(\ddot{s}_h(t + \Delta t)) \]

\[ V_h(t + \Delta t) \leftarrow push(\ddot{v}_h(t + \Delta t)) \]

return \([S_h(t + \Delta t), V_h(t + \Delta t)]\)

2.5 Complexity Analysis

At every step time step, HMBAP executes HMBM to simulate human behaviour. This can significantly add to the computation requirement of the method. Hence it is important to study the time complexity of the algorithm with respect to variables described in table 2.1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_r )</td>
<td>Number of humans in robot’s environment</td>
</tr>
<tr>
<td>( n_h )</td>
<td>Number of entities influencing a human ( h )’s motion behaviour = Number of humans around ( h ) plus 1 robot</td>
</tr>
<tr>
<td>( k )</td>
<td>Number of time steps for which robot plans a path</td>
</tr>
<tr>
<td>( m )</td>
<td>Number of trajectories evaluated at each step</td>
</tr>
</tbody>
</table>

Once the global plan is generated by NF1, for every time step HMBAP compares \( TrCost_i \)
CHAPTER 2. HUMAN MOTION BEHAVIOUR AWARE PATH PLANNER

of $m$ trajectories to obtain the optimal trajectory. To generate $TrCost_i$ for every trajectory corresponding to $\nu_i \in V_r$, the trajectory is discretized into $k$ time steps and at every time step two main routines are executed: 1) Calculate $d_{obs}$ and 2) HMBM.

1) Calculate $d_{obs}$

$d_{obs}$ is the distance of nearest human to the robot. For calculating $d_{obs}$ robot’s euclidean distance is calculated from each human and the minimum is selected. The process is completed in $O(n_r)$ time.

2) HMBM

Human Motion Behaviour Model (HMBM) is used to update the new expected human positions at each time step. The implementation of HMBM is shown in [2]. This process takes $O(n_h)$ time for each $h$. The time required to calculate the position of all $n_h$ humans is $O(n_h^2)$.

The time required for executing these two routines for each time step is $O(n_r) + O(n_h^2)$. For $k$ time steps, the run time required is $O(k \cdot n_r) + O(k \cdot n_h^2)$. For $i$ such trajectories, the time required is $O(m \cdot k \cdot n_r) + O(m \cdot k \cdot n_h^2)$.

Discussion

In the expression for time complexity, $O(m \cdot k \cdot n_r) + O(m \cdot k \cdot n_h^2)$, the value of $k$ depends on the time for which the robot plans a path. A large planning time for the robot gives the robot the ability to plan collision avoidance for a longer period. But on the down side a large planning time results in high computational complexity. Similarly having a large number of evaluated trajectories, $m$, ensures that the robot considers more routes while planning a path; but it adds to the computational complexity. It is thus important to set optimal values for $k$ and $i$ considering the factors above. For our planner we set these as in table A.4.

Once set, the values of $k$ and $m$ are constant and the planning time is largely dependent on $n_r$ and $n_h$. As is seen above the time complexity introduced by HMBM is $O(m \cdot k \cdot n_h^2)$. Thus for high values of $k$ and $n_h$ this term can result in high run time. In order to check this we can make two possible adjustments. First, we assume that at a time only a small number of neighbours (empirically around 5 plus the robot) of human $h$ influence $h$. These neighbours can be chosen according to some criteria based on distance from $h$ or velocity.
direction or a combination of both. This is a reasonable assumption as even in very crowded scenarios, it is unlikely that a human is influenced by more than a small number of humans around it. The second (more simplistic) option is to consider influence on $h$ due to the robot only. This option does not consider other humans influence on $h$ but still ensures that the robot considers its influence on humans while planning a path.
Chapter 3

Simulations

The proposed method has been implemented using ROS (Robot Operating System) [25]. We use Stage for simulation, and RViz for visualization, packages available under ROS. We assume perfect sensing and pass exact human/robot odometry data from Stage to HMBAP. The parameters for dynamics of the robot were set similar to that of Powerbot, the indoor robot. Parameters for HMBM and the TrCost function were set empirically via test simulations. The former were set so that HMBM models appropriate obstacle avoidance behaviour expected of humans. The later were set such that the robot safely avoids obstacles while following the planned global path (See Table A.1).

We have created tough navigation scenarios for the robot which clearly highlight the advantages of our proposed method. The planner was tested in constricted spaces and in crowded environments to test its applicability to different set-ups. To show the advantages of using HMBM in the planner, we first run the simulations without HMBM and then with HMBM. Our results are interesting and encouraging. In this section we explain three of such tested scenarios and their result. The videos for the actual simulation for the three experiments are available as supplemental files with this thesis.

3.1 Simulation setup

3.2 Simulation 1: Human waits for robot to move

The scene for this simulation is as shown in the first (top left) snapshot in column I in Fig. 3.1. The robot is shown in yellow; it is in a hallway and is facing a T junction. A human
obstacle, A, shown in green, is some distance behind the robot and is moving towards it, intending to move to the T-junction. Another human obstacle, B, shown in red, is in left arm of the T junction and is moving towards the T-junction; it intends to continue straight ahead past the T-junction. For all the simulations we use red color for humans that are unaware of the robot, and green color for humans that are aware of the robot. Here human A is aware of the robot and will try to avoid a collision with it, but human B is not aware of the robot’s presence and will not try to avoid a collision. The robot is set a goal in the left arm of the T junction past human B (shown by a blue arrow). The behaviour of the robot is discussed below:

**Without HMBM:** Column I in Fig. 3.1 shows the behaviour of the robot while using the planner without HMBM. NF1 plans a path for the robot which goes to the T-junction and then turns left towards the goal; it is seen in blue in the first sub-figure. The robot moves along this path till it reaches the T junction. The robot has three main alternative paths at this point. The first alternative is to go left along the planned global path, the robot cannot take this path as it expects to collide with human B while executing this path. The second option is to wait at its current location, the robot cannot take this path as it expects a collision with obstacle A on doing so. The third option is to go right, this is a high cost path as it deviates from the planned global path. The robot despite the high cost, takes this third path, as the other two alternatives lead to collision. As seen in the last figure, the robot turns right and continues moving away from its goal to avoid colliding with B. The robot is now trapped in the right arm of the T-junction, as the corridor is wide enough for only one entity to pass.

**With HMBM:** Column II in Fig. 3.1 shows the behaviour of the robot while using the planner with HMBM. As before, the global planner plans a path for the robot which goes to the T-junction and then turns left towards the goal. The robot moves forward to the T junction, and evaluates the three alternative paths available. The robot rejects going left as it expects a collision with obstacle B. The second option is for the robot to wait at its current location; in this case HMBM predicts that if the robot stops, human A will slow down and stop before colliding with the robot, hence the trajectory cost is lowered. This option is still a high cost position for the robot as it is close to obstacle A. The third option is for the robot to go right. This trajectory is a high cost path as it deviates from the global path. The robot takes the second option as it is less costly than a trajectory which deviates considerably from the global path and the goal. The robot waits for B to pass and then
Figure 3.1: Simulation 1 - Human waits for robot to move

I. Without HMBM
II. With HMBM
moves to the goal. This robot behaviour is very human-like and effective in getting the robot to its goal.

### 3.3 Simulation 2: Corridor passing situation

This simulation shows the robot (in yellow) in the center of a hallway, as seen in the first snapshot in Fig. 3.2 column I. From each side of the hallway a human obstacle, shown in green, is walking towards the center of the hallway. We refer to the human in front of the robot as human A and the human behind the robot as human B. The robot is given a goal towards the end of the hallway (shown by a blue arrow), such that it has to pass human A to reach the goal. Both the humans are aware of the robot and will try to avoid a collision with the robot. We discuss the behaviour of the robot below:

**Without HMBM:** Column I in Fig. 3.2 shows the behaviour of the robot while using the planner without HMBM. The global path planned connects the start and goal configuration along the center of the hallway. The robot moves along the global path, but it slows down on reaching close to human A. The robot cannot pass human A, as human A is walking down the center of the hallway leaving no space for the robot to pass on its sides. Since the robot can no longer go forwards, the only other option is to stop or turn around and go back. The robot cannot stop as it expects to collide with the approaching human A. The robot cannot go back either as doing so will lead it to collide with human B. The planner reports that no collision free path can be found. The robot stalls.

**With HMBM:** Column II in Fig. 3.2 shows the robot behaviour. The planned global path is same as before. The robot moves along the global path, but it slows down on reaching close to human A. As before the robot cannot pass human A, as human A is walking down the center of the hallway leaving no space for the robot to pass on its sides. The robot evaluates all possible trajectories. The trajectory moving the robot to one side of the hallway is found to be the lowest cost. The Human Behaviour model estimates that for this trajectory where the robot moves to one side of the hallway, human A will move to the other side of the hallway (since a repulsive force acts on the human forcing it away from the robot). This estimate makes this trajectory a less costly one and the robot executes this trajectory. The robot moves to one side of the hallway, the human moves to the other side of the hallway and the two pass. This is a very interesting behaviour shown by the robot and is similar to what a human would do in such situations.
3.4 Simulation 3: Crowded mall situation

Simulation 3 shows a scene typical in malls where crowds of people move in large courtyards at different speeds. The robot is in one such courtyard as seen in the first snapshot in Fig. 3.3 Column I. In front of the robot is a slow moving human obstacle, say human A. Since the robot is behind A, A is unaware of the robot’s presence. From behind the robot, 3 people are approaching it with relatively faster speed, starting from the left we call them B, C and D. To create an interesting case, human D is considered as distracted, hence unaware of the robot’s presence. B and C on the other hand are aware of the of robot. Two other people are walking, each in front of B and D respectively, and are aware (as indicated by the green color). The robot is given a goal straight ahead past human A (shown by a blue arrow). The robot behaviour is as follows.

**Without HMBM:** Column I in Fig. 3.3 shows the behaviour of the robot. The planned global path is a straight line from the start pose to the goal pose. The robot moves forward along the global path approaching the slow moving person, A; the robot evaluates its options. The first alternative is to overtake the obstacle either from the left or the right
but the robot cannot do this as the path is expected to be in collision with human B or human D, respectively. The robot cannot stay behind human A as human C will be expected to collide with it. The planner reports that it cannot plan a collision free path. The robot stalls.

With HMBM: Column II in Fig. 3.3 shows the behaviour of the robot. The planned global path is the same as before. As the robot moves forward along the global path approaching the slow moving person, A, it evaluates its options. Moving behind A is an option. The robot risks a collision with C but HMBM predicts that C will slow down to avoid a collision with the robot, nevertheless it is a high cost option as the robot is close to C. The second option is to overtake human A from the right; in this case the robot will be in collision with the human D as the model knows D is unaware and will make no effort to accommodate the robot. The third option is to pass human A from the left; in this case the robot risks colliding with B but the HMBM predicts that since B is aware of the robot, it will move further left to avoid a collision with the robot. While both trajectory options, one and three, are collision free, the robot chooses the later as it gets it closer to its goal. The robot takes this trajectory overtaking person A from the left and reaching its goal. The second figure in column II of Fig. 3.3 shows the robot take this path (the figure for this case shows only four humans as the two humans walking on the sides have moved way past the robot’s neighbourhood). This robot behaviour is effective in taking the robot to the goal and is very human-like.
Figure 3.3: Simulation 3 - Crowded mall situation

I. Without HMBM

II. With HMBM

No collision free path available!
Chapter 4

Implementation

This chapter discusses the implementation details of Human Motion Behaviour Aware Planner (HMBAP) on our lab robot. It describes the robot setup, discusses the sensing schemes considered and used, describes methods used for detecting and tracking people and methods used for mapping and localization and finally the integration of different sensing and planning modules.

4.1 Robotic Platform

Figure 4.1 shows the robotic setup. The mobile platform is a Adept MobileRobots’ Powerbot, a high-payload differential-drive platform. The size of the foot-print of the robot is 110cm x 90cm. The manipulator arm on the robot is in the home configuration and is kept in this configuration for all experiments.

Three basic sensors are mounted on the robot. A Microsoft Kinect, RGB-D camera, is mounted to the front of the robot at an height of 60cm from the ground. A Sick LMS1 laser range finder is mounted on the front of the robot at a height of 22 cm from the ground. And lastly a Hokuyo laser range finder is mounted at the back of the robot at a height of 15 cm from the ground.

4.2 Detecting and tracking people

The first step to avoiding collisions with people is reliably detecting and tracking people. For implementing HMBAP it was important to come up with a sensing scheme that can
Figure 4.1: Robotic platform
achieve this goal. In [27] Trautman et al use overhead cameras to track people. Using overhead cameras significantly simplifies the tracking task as a simple background subtraction technique can accurately detect human positions. Despite its obvious advantage, an overhead camera sensing approach is not suitable as it significantly limits the robot’s workspace to the area watched by the camera. A more general approach is to use sensors on-board the robot.

Tracking people from a mobile platform is a non-trivial problem. For one, simple background subtraction techniques can no longer be used as the background constantly changes as the robot moves around. Another problem that develops from this is the need for accurate mapping of the robot’s environment and accurate localization of the robot in this map. All measurements taken by the sensors on-board the robot are with respect to the robot’s current pose, any error in the pose of the robot reflects on the measurements. The biggest problem of using on-board sensors is that of occlusion. Humans which are occluded by static obstacles or other humans are not easily detected. Also humans walking in and out of line of sight of the sensors make data associations complicated.

**Object tracking with Kinect**

The first sensing method for tracking people was considered using the Microsoft Kinect mounted to the front of the robot. The depth image from the Kinect gives a 640x480 pixel image where each pixel intensity value directly corresponds to its distance from the camera. The advantage of using a depth image is that separating the background from the foreground becomes easy. We used a region growing approach, introduced in [28] to segment out objects of interest.

Starting from the pixel with the highest value (representing the point closest to the robot) we plant seeds in the image and using a similarity measure $S$ recursively classify neighbours of the seeded pixels as belonging to the same cluster. A similarity $S$ between a cluster pixel $x$ and a neighbouring pixel $y$ is defined as:

$$S(x, y) = |\mu_x - D_y|$$

(4.1)

Where, $D_y$ is the distance value of pixel $y$ and $\mu_x$ is a local parameter defined as follows. When a seed is planted, $\mu_x$ is initialized to $D_x$. When a neighbour $y$ of seed $x$ is absorbed, $\mu_y$ is computed as follows:

$$\mu_y = \frac{\mu_x \cdot n + D_y}{n + 1}$$

(4.2)
Here \( n \) is number of neighbours the pixel \( y \) has in the cluster. \( y \) is absorbed into the cluster if \( S(x, y) < \theta \) where \( \theta \) is a constant threshold (listed in table A.2). Clusters with dimensions comparable to humans (table A.2) were retained while the rest discarded. These clusters were represented by a uniquely coloured 3 dimensional bounding box across the extremities of the segments. The centroid of the cluster was also recorded along with the bounding box. Correspondence was established among clusters across multiple frames based on the distance between their centroids and the overlap in their bounding boxes in consecutive frames.

Detecting humans using a Kinect in this way proved to be sufficiently robust and fast. But one of the real issues of using a Kinect, is its small horizontal field of view of 57 degrees. In order to work in a dynamic environment the robot needed to track multi objects in a large, preferably 360 degree field of view. The Kinect can sense a small window of this field of view at a time, thus tracking multiple objects is not possible unless the Kinect itself is panned at a high rate. We opted for a simpler and less costly solution for now, as explained next.

**Tracking humans with 2D laser range finders**

Commercially available 2D Laser Range Finders (LRFs) or line scanners have a large field of view. Two LRFs, one front and one back, can give the robot a field of view of near 360 degrees. The problem with LFRs is that being line scanners, they scan a single (in our case, horizontal) slice of the scene. For our application the LRFs are located at about 15cm to 25cm from the ground, thus if a human walks in its field of view, all it sees is a line scan of the lower legs of the person.
CHAPTER 4. IMPLEMENTATION

Detecting Humans

Fortunately, it is possible to detect/track humans by detecting leg like shapes in a laser scan. Arras et al [29] use boosted features to detect humans. The scan from the range sensor consists of a set of beams emanating from the scanner at a resolution of 0.5 degrees. The value corresponding each beam represents the distance of an object relative to the robot. The beams in the scan are split into subsets of beams based on a segmentation algorithm. A jump distance condition is used to compute the segmentation: If two adjacent beams are farther away than a threshold distance, a new subset is initialized. Fourteen features are defined for each of these segments based on geometry. Ada-boost [30] was used to learn a strong classifier using these features. The training sets were segments labelled manually as person or non-person. The best features for detecting legs is given by the importance of the individual feature weights in the final strong classifier. Table 4.1 lists features with the highest weights.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>Radius of the circle fitted to the segment</td>
</tr>
<tr>
<td>Mean angular difference</td>
<td>This features is a measure of the convexity or concavity of segment (defined in eq. A.1)</td>
</tr>
<tr>
<td>Jump distance to adjacent segments</td>
<td>Euclidean distance between the current segment and its neighbours</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>Standard deviation from the mean for the points in the segment</td>
</tr>
</tbody>
</table>

Tracking Humans

Once we can detect legs we set up Kalman filters to track them. A leg track is represented as $\mathbf{x} = (x, y, v_x, v_y)$ where $x$ and $y$ are track positions and $v_x$ and $v_y$ are $x$ and $y$ components of the track velocity. For motion prediction, a constant velocity model is employed. A key problem in tracking is that of data association as a person’s legs are often occluded by each other. In order to achieve robust data association, a Multi Hypothesis Tracker (MHT) is used as described by Reid [31].

Multi Hypothesis Tracker uses Bayesian learning to learn the probability of hypothesis $\Omega_j^k$ given measurements $z_k$. $z_k$ is complete LRF scan at time $k$. It defines $\Psi_j(k)$ which
associates each measurement either to an existing track, a false alarm (denoted as \( fa \)) or a new track (denoted as \( nt \)) and marks a track as detected or deleted. Table 4.2 shows an example of such assignments. Here, track \( x_1 \) is assigned to measurement \( z_2 \), track \( x_2 \) is scheduled for deletion, and measurement \( z_1 \) is set as new track. There are as many possible assignment sets \( \Psi_j(k) \) as we can distribute 1’s and 0’s associating tracks to measurements. Given probability of previous hypothesis \( \Omega_k^{j-1} \) and \( \Psi_j(k) \), under assumptions given in [31], we can calculate the probability of \( \Omega_j^k|z_k \):

\[
p(\Omega_j^k|z_k) = p(\Psi_j(k), \Omega_k^{j-1}|z_k)
= \eta p(z_k|\Psi_j(k), \Omega_k^{j-1})p(\Psi_j(k)|\Omega_k^{j-1})p(\Omega_k^{j-1})
\]

(4.3)

Table 4.2: Example for set of assignments

<table>
<thead>
<tr>
<th></th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_{nt} )</th>
<th>( x_{fa} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z_1 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( z_2 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( z_{del} )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Here, \( \eta \) is the normalizer, the term \( p(z_k|\Psi_j(k) \) is the measurement likelihood, \( p(\Psi_j(k)|\Omega_k^{j-1}) \) is the probability of an assignment set, and \( p(\Omega_k^{j-1}) \) is the probability of hypothesis \( \Omega_j^{k-1} \), which has been computed in the previous iteration. In order to obtain \( p(\Psi_j(k)|\Omega_k^{j-1}) \). Reid [31] classifies each track measurement as detected or not detected and obtains an easy to calculate expression for probability of assignment in terms of \( N_{det}, N_{fal}, N_{new} \), which are the number of measurements associated with detected targets, false targets and new targets, respectively.

In order to account for the higher probability of legs of a human being occluded by each other, the MHT expression is modified as explained by Arras et al[32]. We associate legs track to person tracks where a person track is defined as a high-level track to which two legs tracks are associated using the following model for a person track:
1) People have two legs
2) Legs are close to each other
3) Legs move in a similar direction
4) Legs are more likely to occlude each other than be occluded by other objects
5) The state of a person is estimated from the state of the two legs tracks using the multivariate weighted mean. Leg tracks are thus either associated to a person track or to no person track at all, referred to as free tracks. Let $N^A$ represent tracks associated to person tracks and $N^F$ represent free tracks. At any time instant leg tracks can be either detected, occluded or not detected (to be deleted), with probability $p_{\text{det}}$, $p_{\text{occ}}$ and $p_{\text{del}}$ respectively. Separating the probabilities for associated tracks $N^A$ and free tracks $N^F$ we define probabilities as $p_{\text{det}|A}$, $p_{\text{occ}|A}$ and $p_{\text{del}|A}$ for $N^A$ tracks and $p_{\text{det}|F}$, $p_{\text{occ}|F}$ and $p_{\text{del}|F}$ for $N^F$ tracks. Using operations and assumptions as in [31] an expression for child hypothesis is obtained as:

$$p(\Omega^k_j | z_k) = \eta^M_k \prod_{i=1}^{M_k} N(z_i^k)^\delta \cdot p_{\text{det}}^{N^F} \cdot p_{\text{occ}}^{N^F} \cdot p_{\text{del}}^{N^F} \cdot p_{\text{det}}^{N^A} \cdot p_{\text{occ}}^{N^A} \cdot p_{\text{del}}^{N^A} \cdot \lambda_{\text{new}}^{N_{\text{new}}} \cdot \lambda_{\text{fal}}^{N_{\text{fal}}} \cdot p(\Omega^{k-1}_p)$$ (4.4)

Here $\lambda_{\text{new}}^{N_{\text{new}}}$, $\lambda_{\text{fal}}^{N_{\text{fal}}}$ represent average rate of events per unit volume for $N_{\text{new}}$ and $N_{\text{fal}}$ respectively assuming they follow a Poisson distribution. For the complete derivation of the final expression please refer to [32]. This segregation gives us the option to express conditional probability for associated tracks and free tracks differently. For instance, we set $p_{\text{occ}|A}$ higher than $p_{\text{occ}|F}$ in order to express the higher probability of legs of a human being occluded by each other.

Table A.5 lists all the parameters used for people tracking. This method was tested to give robust people tracking by the means of leg detection. People can be detected and tracked in almost 360 degree field of view to a distance of 5 meters from the robot. At distances greater than 5 meters laser scanner beams become to sparse to capture the shape of human legs. This method was adopted for extracting people position and velocity information for the planner implementation. The leg detector node under ROS provides the implementation for this node.

4.3 Localization and mapping

Accurate mapping and localization is very important especially when the robot relies on onboard sensors for sensing its environment. We use a Rao-Blackwellized particle filter based method for Simultaneous Localization and Mapping (SLAM) explained by Montemerlo et al [33]. This particle based filter outperforms earlier Extended Kalman Filter (EKF) based methods in computational complexity and robustness against incorrect data associations.
CHAPTER 4. IMPLEMENTATION

The key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(x_{1:t}, m|z_{1:t}, u_{1:t})$ about the map $m$ and the robot trajectory $x_{1:t}$. Using conditional independences mentioned in [35], we can write this as:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(m|x_{1:t}, z_{1:t}) \cdot p(x_{1:t}, u_{1:t-1})$$  \hspace{1cm} (4.5)

This factorization allows us to first estimate only the trajectory of the robot and then to compute the map given that trajectory. While $p(m|x_{1:t}, z_{1:t})$ can be computed analytically, we use a particle filter to estimate $p(x_{1:t}, u_{1:t-1})$. We define a particle $x_{1:t}^{(i)}$ such that each particle represents a potential trajectory of the robot. The RBPF process of estimating the posterior is summarized below:

1) **Sampling**: The next generation of particles $\{x_{t}^{(i)}\}$ is obtained from the generation $\{x_{t-1}^{(i)}\}$ by sampling from the proposal distribution $\pi$.

2) **ImportanceWeighting**: Each particle sampled in step 1 is assigned a weight $w_{t}^{(i)}$ to incorporate the likelihood of sensor measurement for each sample. It is defined as:

$$w_{t}^{(i)} = \frac{\text{targetdist.}}{\text{proposaldist.}} \cdot \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})}$$  \hspace{1cm} (4.6)

The choice of proposal distribution greatly improves SLAM results in RBPF. Grisetti et al. [34] show that in cases where the sensor information is much more precise than odometry information, choosing a proposal distribution based on latest observed sensor data results in superior performance. The weights in this case are evaluated as:

$$w(i)_{t} = w(i)_{t-1} \cdot \int p(z_{t}|x^{'}) p(x^{'}|x_{1:t-1}^{(i)}, u_{t-1}) dx'$$  \hspace{1cm} (4.7)

These weights are recursively calculated and assigned to the samples.

3) **Resampling**: Particles are drawn with replacement proportional to their importance weight. This ensures that particles that support current sensor measurement over other particles which do not.

4) **MapEstimation**: For each particle, the corresponding map estimate $p(m^{(i)}|x_{1:t}^{(i)}, z_{1:t})$ is computed based on the trajectory $x_{1:t}^{(i)}$ of that sample and the history of observations $z_{1:t}$.

The gmapping package from ROS was used for the implementation of the above method. Table A.3 lists all parameters used for the gmapping node. Figure 5.1 shows the map of Engineering Science Lab at Simon Fraser University as created using the method above. Mapping was done by manually moving the robot in the region and sensing using the front LMS 100 laser sensor.
4.4 Planner implementation in ROS

HMBAP is implemented on the robot under ROS [25]. Figure 4.4 shows all the different software modules (nodes) run by HMBAP and the interaction between them. While the sensing and control nodes run on the Powerbot, the planning nodes run on the lab laptop. The laptop is powered by a 2.5-GHz quad-core intel processor and 8 Giga Bytes of RAM. The nodes communicate with each other through messages by publishing and subscribing to topics. A message is a custom data structure which contains the data being published/ subscribed on a particular topic. We describe all the nodes and list all the parameters for all these nodes in Appendix A.
The planning node is implemented through the `move_base` node. `move_base` receives a goal pose from the user and runs HMBAP to generate velocity commands for the robot. `move_base` receives people position and velocity measurements from the `people_pos_vel_generator` node and the environment map from `gmapping`. The velocity commands generated by `move_base` are subscribed to by `p2os` and passed to the drive system of the robot. For a more detailed description of each node see Appendix A.
Figure 4.4: Directed graph showing all nodes and the communication links between them.
Chapter 5

Experiments

HMBAP has been tested with simulated humans which move according to the Human Motion Behaviour Model. HMBM captures the main motion objectives of humans but in reality we do not expect people to behave exactly as described by HMBM. Thus testing HMBAP on a robot around real humans verifies its effectiveness. In order to ensure safety of human subjects in our experiments, we added an additional condition in HMBM, that is, the predicted velocity of a human cannot become negative (it cannot have a velocity component in the direction opposite to it’s desired velocity). This condition ensures that the robot does not come too close to people expecting them to move back.

In this section we describe the experiments conducted on the real robot. We run the planner in complex navigation situations and show the advantage of HMBM by first running the planner without HMBM and then with HMBM. The videos of the experiments are available as supplemental files with this thesis.

5.1 Experimental setup

We tested the method on our lab robot, described in section 4.1, in narrow corridors around RAMP lab at SFU. A world map for the lab environment was built using the methods described in section 4.3. Figure 5.1 shows the generated world map used in the experiments. Front and back laser line scanners were used to detect and track people in the lab as explained in 4.2. We present here two tough navigation situations in which the knowledge of reaction of humans to robot’s motion helped the robot plan a suitable path to reach its goal. First, we consider a situation where the robot passes a human in a narrow corridor (figure 5.3).
Second, we consider a situation where the robot is navigating a 4-way crossing in presence of moving humans (figure 5.5). In all the examples the human subjects are given a starting point and a goal and are told to avoid the robot as they would avoid collisions with other people.

5.2 Experiment 1: Passing a human in a corridor

In this experiment we test the behaviour of the robot while passing a human in a narrow corridor. The world map for this experiment is shown in fig 5.1 (a). The robot is at one end of the hallway and is given a goal at the other end shown by the red flag. From each side of the hallway a human, as seen in figure 5.2, walks towards the center of the hallway. We refer to the human in front of the robot as human A and the human behind the robot as human B. In order to reach the goal the robot has to pass human A on the side while avoiding B. Both the humans are assumed to be aware of the robot and will try to avoid a collision with the robot. We discuss the behaviour of the robot below.
Without HMBM

Fig. 5.2 shows the behaviour of the robot while using the planner without HMBM. The global path planned connects the start and goal configuration along the center of the hallway. The robot moves along the global path, but it slows down on reaching close to human A. The robot cannot pass A, as human A is walking down the center of the hallway leaving no space for the robot to pass on its sides. Since the robot can no longer go forwards, the only other option is to stop or turn around and go back. The robot cannot stop as it expects to collide with the approaching human A. The robot cannot go back either as doing so will lead it to collide with human B. The planner reports that no collision free path can be found. The robot stalls.

With HMBM

Fig. 5.3 shows the robot behaviour. The planned global path is same as before. The robot moves along the global path, but it slows down on reaching close to human A. As before the robot cannot pass human A, as human A is walking down the center of the hallway leaving no space for the robot to pass on its sides. The robot evaluates all possible trajectories. The trajectory moving the robot to one side of the hallway is found to be the lowest cost. The Human Behaviour model estimates that for this trajectory where the robot moves to one side of the hallway, human A moves to the other side of the hallway (since a repulsive force acts on the human forcing it away from the robot). This estimate makes this trajectory a less costly one and the robot executes this trajectory. The robot moves to one side of the hallway, the human moves to the other side of the hallway and the two pass. This is a very interesting behaviour shown by the robot and is similar to what a human would do in such situations.

5.3 Experiment 2: Robot at a 4-way crossing

In this experiment the robot is given a goal such that it has to navigate a 4-way crossing. In this case, occlusions can play a major role as the robot has no idea of people approaching the crossing since they are occluded from the robot’s line of sight. The world map for this experiment is shown in fig 5.1 (b). The robot is in the hallway facing the 4-way crossing and

\[1\text{Robot report no collision free trajectories and stops}\]
Figure 5.2: Experiment 1: Passing a human in a corridor - Without HMBM
CHAPTER 5. EXPERIMENTS

is given a goal to the left of the 4-way crossing shown by the red flag. A human obstacle, (say A), is some distance behind the robot and is moving towards it; A is given a goal past the 4-way crossing. Another human (say B) is approaching the 4-way crossing from the left and is also given a goal straight ahead (see figure 5.4). In order to reach the goal, the robot has to pass human A on the side while avoiding B. While A is assumed to be aware of the robot, B is assumed to be unaware of the robot as the robot is occluded from B. The behaviour of the robot is discussed below:

Without HMBM

Fig. 5.4 shows the behaviour of the robot while using the planner without HMBM. NF1 plans a path for the robot which goes to the 4-way crossing and then turns left towards the goal. The robot moves along this path till it reaches the crossing. The robot now sees B coming towards it. The robot has three main alternative paths at this point. The first alternative is to go left along the planned global path, the robot cannot take this path as it expects to collide with human B while executing this path. The second option is to wait at its current location, the robot cannot take this path as it expects a collision with human A on doing so. The third option is to go right, unlike in the simulations under similar conditions, this path is not collision-free for the robot. The reason for this is occlusions, the robot sees B at such a stage that it cannot take a right turn in time to be able to avoid a collision. As seen in the last frame in fig. 5.4 the robot sees no collision free paths and stops in its tracks.

With HMBM

Fig. 5.5 shows the behaviour of the robot while using the planner with HMBM. As before, the global planner plans a path for the robot which goes to the 4-way crossing and then turns left towards the goal. The robot moves forward to the crossing, and detects B approaching the crossing. The robot evaluates three alternative paths available. The robot rejects going left as it expects a collision with obstacle B. The second option is for the robot to wait at its current location; in this case HMBM predicts that if the robot stops, human A will slow down and stop before colliding with the robot, hence the trajectory cost is lowered. This

2Robot uses HMBM to find a path
3Robot report no collision free trajectories and stops
Figure 5.3: Experiment 1: Passing a human in a corridor - With HMBM
CHAPTER 5. EXPERIMENTS

Figure 5.4: Experiment 1: Robot at crossing - Without HMBM
option is still a high cost position for the robot as it is close to obstacle A. The third option is for the robot to go right. This trajectory is a high cost path as it deviates substantially from the global path. The robot takes the second option as it is less costly than a trajectory which deviates considerably from the global path and the goal. The robot waits for B to pass and then moves to the goal. This robot behaviour is very human-like and effective in getting the robot to its goal.
Chapter 6

Conclusion and future work

6.0.1 Conclusion

We introduced a Human Motion Behaviour Aware Planner (HMBAP) that takes into account of how people react to the robots planned motion while planning it’s motion. The planner uses the DWA to generate admissible trajectories, and then uses a Human Motion Behaviour Model (HMBM) to estimate how a human would move in response to each of these trajectories. The predicted human trajectories are taken into account to evaluate the robot trajectories and choose the lowest cost one. We extend our method to incorporate the distinction of aware (they have seen the robot and will change their motion in response to robot’s motion, if needed, to avoid collision) or unaware (they are distracted or have not seen the the robot, and hence will continue along their path even if it results in collision) people. The method is fast and shows a human like behaviour of the robot in avoiding other humans around it.

In initial tests, HMBAP was shown to work well in real human environment where humans avoid the robot as they avoid other humans. A comparison of the runs of the navigation algorithm with and without the human motion behaviour model shows the advantage of incorporating the model in the planner. The implementation of the method using on-board line scanners (LRFs) to track people and estimate their velocities resulted in some limitations. One, people could not be detected more than 5 meters from the robot, as a very small number of scans were available off their legs. Second, with lasers in the front and back of the robot, the robot had complete vision front and back, but it had blind regions on its sides. This made conducting experiments in a crowed human environments risky. Other
method of achieving better sensing were discussed and are presented as future work.

6.0.2 Future work

Integration with hierarchical safety approach

The underlying method used for trajectory generation and evaluation in our Human Motion Behaviour Aware Planner (HMBAP) is a Dynamic Window based method. While the use of Dynamic Window has proved to be effective for navigation in dynamic human environment, a more comprehensive method is needed for improved safety. For example, a robot exiting a hallway must take into consideration the possibility of moving obstacles around the corner of the hallway and catching the robot in an odd spot. Another situation is where people might be occluded from the vision of the robot by the presence of other people and might appear unexpectedly near the robot. Sometimes, people trajectory estimates received by the robot can be unreliable and this might result in unsafe behaviour from the robot. The robot should plan for such contingencies for safer motion planning among humans. Towards this goal, a hierarchical safety approach is being developed by Bruno L’esperance, a Ph.D. student in the RAMP Lab and is reported in [36]. This approach accounts for potential risk scenarios for the robot occurring due to occlusions or unforeseen changes in the environment. We plan to incorporate HMBAP within this safety hierarchy approach to develop a comprehensive safety planner.

Sensing scheme

One of the limitations of this work was the limited sensing capability. While laser range finders have been used here to track people, the single line scan limitation warrants a better sensor scheme. One solution is to couple the LRF sensors with a Kinect like sensors to get better estimates about a person’s trajectory. While this scheme gives some improvement, it is again limited in the sense that the Kinect can only focus on some targets at a given time. In order for a robot to navigate around people, the robot needs to continuously track humans in a 360 degree field of view. One solution to sensing requirement is a 3D LiDAR like a Velodyne HDL-32E. This sensor provides depth data in a 360 degree field of view at a rate of 15Hz. Currently we are exploring methods to robustly detect and track people in 3D LiDAR data from a Velodyne HDL-32E. The improved sensing will ensure safer navigation behaviour of the robot in crowded scenarios.
Classification of humans as *aware* or *unaware*

While incorporating influence of robot’s planned trajectory on future human trajectories is advantageous (as demonstrated in this work), it is important to determine whether a human is *aware* or *unaware* of the robot. In this work pre-determined flag values are assumed, but future research in determining these values from observed sensor data, will greatly benefit HMBAP in successfully navigating human environments. As mentioned in the section 2 this classification could be based on visual clues, by determining a person’s field of view or gaze direction or be inferred from tracking a person’s trajectory. Also in place of a hard classification a confidence measure could associated to each person being *aware* of the robot. The cost of distance from people can then be associated to this value. This will result in the robot behaviour where the robot maintains a greater distance from people who are more likely to be *unaware* of the robot’s presence.

Another issue worth exploring is how the accuracy of this classification routine affects robot behaviour. If a person classified as *aware* by the routine ends up being *unaware* of the robot, how does it affect robot navigation. Initial investigations and simulation of this situation suggest that in these cases the robot gets itself in situations where it can find no collision free paths and it issues a stop command.
Appendix A

ROS nodes used in planner implementation

ROS Master (rosout)
The ROS Master provides naming and registration services to the rest of the nodes in the 
ROS system. It tracks publishers and subscribers to message topics. It enables individual 
ROS nodes to locate one another. Once these nodes have located each other they commu-
nicate with each other peer-to-peer. We run the ROS master on the laptop. The nodes 
runtime on the Powerbot are also registered under this node.

hokuyo_node
hokuyo_node runs the driver for the Hokuyo LRF located at the back of the robot. This 
node publishes sensor data to the topic base_scan at the rate of 10Hz. base_scan is then 
subscribed to by the leg_detector node which uses the data for people detection/tracking.

sickLMS_node, laser_filter, throttle_LMS
sickLMS_node runs the driver for the Sick LMS100 LRF located at the front of the robot. 
This node publishes sensor data to the topic scan at the rate of 25Hz. The field of view of 
the LRF is 270 degrees and this results in parts of the robot’s body entering the FOV of the 
sensor. These sensor readings corresponding to the robot’s body are filtered out from scan 
by the laser_filter node and the filtered scan is published as LMS_scan. LMS_scan is then 
subscribed to by the throttle_LMS node which published the data as base_scan at a lower 
frequency of 10Hz. Publishing frequency of the sensor data from the sick LRF is reduced 
to 10Hz to match that of the Hokuyo LRF.
P2OS
This node runs the driver for the Powerbot. The node subscribes to the topic cmd_vel, which are velocity commands generated by the planner node. P2OS moves the robot in accordance with the linear and angular velocity values contained in cmd_vel. While moving the robot, P2OS also publishes odometry information about the robot’s pose by publishing the transformation matrix between the current pose of the robot and it’s starting pose.

robot state publisher
This node publishes the transformation matrices (tf) for the frames attached to the sensors onboard the robot with respect to the frame attached to the body of the robot.

leg_detector
The leg_detector runs the implementation of people detection/tracking discussed in section 4.2. This node subscribes to base_scan and publishes people_tracker_measurements which are people location measurements.

people_pos_vel_publisher
This node subscribes to people_tracker_measurements and builds a list of people present in the robots environment. By recording people location measurements over time it calculates velocity vectors for all people present. The node then publishes the list of positions and velocities of these people to the topic OdometryMovingObstacles which is subscribed to by move_base.

gmapping This node runs the Simultaneous Localization And Mapping (SLAM) method discussed in section 4.3. This node subscribes to data from the sick LMS LRF, via the topic LMS_scan and publishes the workspace map to the topic map. gmapping publishes localization data by publishing the transformation matrix between the world frame map and robot’s starting pose odom.

move_base
The move_base node subscribes to topic move_base/goal which is the goal pose given to the robot by the user. move_base then runs an implementation of algorithm 1 to generate velocity commands and publishes them to cmd_vel. cmd_vel is then subscribed to by P2OS
which then moves the robot according to these velocity commands.

**HMBAP Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Robot Dynamics</strong></td>
<td></td>
</tr>
<tr>
<td>Maximum linear acceleration</td>
<td>$\dot{\theta}_{\text{max}} = 2 \text{m/s}^2$</td>
</tr>
<tr>
<td>Maximum linear velocity</td>
<td>$\theta_{\text{max}} = .4 \text{m/s}$</td>
</tr>
<tr>
<td>Maximum angular acceleration</td>
<td>$\omega_{\text{max}} = 5 \text{rad/s}^2$</td>
</tr>
<tr>
<td>Maximum angular velocity</td>
<td>$\omega_{\text{max}} = 1 \text{rad/s}$</td>
</tr>
<tr>
<td><strong>HMBM parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Relaxation time</td>
<td>$\tau_h = .07 \text{s}$</td>
</tr>
<tr>
<td>Weight for repulsive force</td>
<td>$w^{\text{rep}} = 1$</td>
</tr>
<tr>
<td>Threshold distance for repulsive force</td>
<td>$d_{\text{thresh}} = 3 \text{m}$</td>
</tr>
<tr>
<td><strong>TrCost parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Weight on goal distance</td>
<td>$\alpha = 24$</td>
</tr>
<tr>
<td>Weight on path distance</td>
<td>$\beta = 32$</td>
</tr>
<tr>
<td>Weight on cost due to obstacles</td>
<td>$\gamma = 2.53$</td>
</tr>
<tr>
<td>Obstacle cost scaling factor</td>
<td>$\eta = 10$</td>
</tr>
</tbody>
</table>

| Table A.1: Parameters values for HMBAP |

**Parameters for Kinect tracking**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity threshold</td>
<td>$\theta = 3 \text{ pixels}$</td>
</tr>
<tr>
<td>Human dimensions</td>
<td>$100 &lt; \text{width} &lt; 300 \text{ (pixels)}$</td>
</tr>
</tbody>
</table>

| Table A.2: Parameters for detecting humans using Kinect |
### Mean angular difference

This feature traverses the boundary of the segment and calculates the average of the angles \( \beta_j \) between the vectors \( \mathbf{x}_{j-1} \mathbf{x}_j \) and \( \mathbf{x}_j \mathbf{x}_{j+1} \) where \( x_i \) is the \( i^{th} \) point on the segment and,

\[
\beta = \angle (\mathbf{x}_{j-1} \mathbf{x}_j, \mathbf{x}_j \mathbf{x}_{j+1}) \quad (A.1)
\]

### gmapping parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverted_laser</td>
<td>false</td>
<td>maxUrange</td>
<td>30.0</td>
</tr>
<tr>
<td>particles</td>
<td>30</td>
<td>delta</td>
<td>0.10</td>
</tr>
<tr>
<td>xmin</td>
<td>-15.0</td>
<td>xmax</td>
<td>15.0</td>
</tr>
<tr>
<td>ymin</td>
<td>-15.0</td>
<td>ymax</td>
<td>15.0</td>
</tr>
<tr>
<td>angularUpdate</td>
<td>0.5</td>
<td>linearUpdate</td>
<td>1.0</td>
</tr>
<tr>
<td>map_update_interval</td>
<td>1.0</td>
<td>resampleThreshold</td>
<td>0.3</td>
</tr>
<tr>
<td>ll_samplerange</td>
<td>0.05</td>
<td>llsamplestep</td>
<td>0.05</td>
</tr>
<tr>
<td>las_samplerange</td>
<td>0.05</td>
<td>lasamplestep</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table A.3: gmapping parameters

### move_base parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local costmap parameters</td>
<td></td>
<td>Global costmap parameters</td>
<td></td>
</tr>
<tr>
<td>global_frame</td>
<td>odom</td>
<td>global_frame</td>
<td>map</td>
</tr>
<tr>
<td>robot_base_frame</td>
<td>base_link</td>
<td>robot_base_frame</td>
<td>base_link</td>
</tr>
<tr>
<td>update_frequency</td>
<td>1.0</td>
<td>update_frequency</td>
<td>1.0</td>
</tr>
<tr>
<td>publish_frequency</td>
<td>0.5</td>
<td>publish_frequency</td>
<td>0.5</td>
</tr>
<tr>
<td>static_map</td>
<td>false</td>
<td>static_map</td>
<td>true</td>
</tr>
<tr>
<td>rolling_window</td>
<td>true</td>
<td>rolling_window</td>
<td>false</td>
</tr>
<tr>
<td>width</td>
<td>5.0</td>
<td>width</td>
<td>60.0</td>
</tr>
<tr>
<td>height</td>
<td>5.0</td>
<td>height</td>
<td>60.0</td>
</tr>
<tr>
<td>resolution</td>
<td>0.05</td>
<td>resolution</td>
<td>0.05</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>-------------------</td>
<td>-------</td>
</tr>
<tr>
<td>origin_x</td>
<td>0.0</td>
<td>origin_y</td>
<td>-30.0</td>
</tr>
<tr>
<td>Planner parameters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transform_tolerance</td>
<td>2.0</td>
<td>world_model</td>
<td>costmap</td>
</tr>
<tr>
<td>sim_time</td>
<td>4.0</td>
<td>sim_granularity</td>
<td>0.025</td>
</tr>
<tr>
<td>vx_samples</td>
<td>5</td>
<td>vtheta_samples</td>
<td>20</td>
</tr>
<tr>
<td>max_vel_x</td>
<td>.3</td>
<td>min_vel_x</td>
<td>0.05</td>
</tr>
<tr>
<td>max_trans_vel</td>
<td>.4</td>
<td>min_trans_vel</td>
<td>0.05</td>
</tr>
<tr>
<td>max_rot_vel</td>
<td>1</td>
<td>min_rot_vel</td>
<td>0</td>
</tr>
<tr>
<td>xy_goal_tolerance</td>
<td>0.12</td>
<td>yaw_goal_tolerance</td>
<td>0.2</td>
</tr>
<tr>
<td>goal_distance_bias</td>
<td>24</td>
<td>path_distance_bias</td>
<td>32</td>
</tr>
<tr>
<td>occdist_scale</td>
<td>.05</td>
<td>forward_point_dist</td>
<td>0.325</td>
</tr>
<tr>
<td>acc_lim_th</td>
<td>5</td>
<td>acc_lim_x</td>
<td>2</td>
</tr>
<tr>
<td>controller_frequency</td>
<td>10</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Table A.4: move_base parameters

**leg_detector parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed_frame</td>
<td>odom</td>
<td>connection_threshold</td>
<td>.3</td>
</tr>
<tr>
<td>min_points_per_group</td>
<td>3</td>
<td>leg_reliability_limit</td>
<td>.3</td>
</tr>
<tr>
<td>leg_pair_separation</td>
<td>.7</td>
<td>max_second_leg_age</td>
<td>1</td>
</tr>
</tbody>
</table>

Table A.5: leg_detector parameters
Bibliography


