# Mobile Robots in Human Environments

*towards safe, comfortable and natural navigation*

Svenstrup, Mikael

Mikael Aslak Svenstrup

# *Mobile Robots in Human Environments*

*towards safe, comfortable and natural navigation*

AALBORG UNIVERSITY

Mobile Robots in Human Environments;
towards safe, comfortable and natural navigation
Ph.D. thesis

ISBN: 978-87-92328-36-6
June 2011

Cover page: a photo of me and our little robot, Robotino

# Preface and Acknowledgements

The thesis in front of you is submitted as a plurality of papers[1] in partial fulfilment of the requirements for Doctor of Philosophy at the Section of Automation and Control, Department of Electronic Systems, Aalborg University, Denmark. The work has been done during the period September 2007 to April 2011 under the supervision of Professor Thomas Bak and Associate Professor Hans Jørgen Andersen.

First and foremost, I would like to acknowledge Thomas Bak for his support in technical discussions and for helping setting the path during the project. I would also like to acknowledge Hans Jørgen Andersen for his inspiration and creative ideas. I would also like to thank Simon and Carsten for their work done during their master's thesis project, which inspired the initial path for this Ph.D. project.

Then I would like to thank Søren Tranberg Hansen for valuable discussions, cooperation and help, especially when the robot did not do what we wanted it to do. Of course I should also thank Robotino, our faithful little robot with the happy face, who is basis for most of the experimental work. Thanks are also due to the all colleagues at the section of Automation and Control, for making it a very pleasant working environment.

During the autumn 2009, I was a guest at University of Nevada in Reno (UNR). There, I would like to thank everybody for making it a very pleasant stay, and especially Monica Nicolescu for making the visit possible.

Finally I would like to thank my wife Janne for supporting me, and especially for your patience during the periods, where I have just been staring at my computer monitor.

Aalborg University
June 2011
Mikael Svenstrup

[1]A short description of the term plurality of papers can be found here:
http://phd.ins.aau.dk/intranet/templates/3863476

# Abstract

Traditionally, robots have been assistant machines in factories and a ubiquitous part of science fiction movies. But within the last decade the robots have started to emerge in everyday human environments. Today they are in our everyday environment in the shape of, for example, vacuum cleaners, lawn mowers, toy pets, or as assisting technologies for care giving. If we want robots to be an even larger and more integrated part of our everyday environments, they need to become more intelligent, and behave safe and natural to the humans in the environment. This thesis deals with making intelligent mobile robotic devices capable of being a more natural and sociable actor in a human environment. More specific the emphasis is on safe and natural motion and navigation issues.

First part of the work focus on developing a robotic system, which estimates human interest in interacting with the robot. This information is then used to navigate appropriately in the environment around a person. The system consists of three parts; finding the person's state information (position, velocity and orientation) relative to the robot, determining if the person wants to interact with the robot, and finally human-aware navigation that respects the persons social zones and interest in interaction. From laser range scanner measurements, a new Kalman filter based method is used to infer the person's state information. Secondly, the robot adaptively learns to estimate if a person seeks to interact with the robot. This is done using a Case-Based Reasoning System (CBR), which analyses current behaviour patterns while comparing these to experiences and outcomes from previous human encounters. The motion of the robot is controlled based on adaptive potential functions. The potential functions are adjusted in accordance with the current interest in interaction and such that the person's social spaces are respected. The operation of the system is evaluated in an open hall setting at the university. It is demonstrated, that the robot is able to learn where to position itself, and is capable of adapting to a changing environment. It is then shown that the developed framework can be used in other applications as well. The framework is transformed into a robotic game that, for example, can be used to motivate elderly people to a regular amount of exercise, which then keep them in better physical shape.

As well as being able to navigate safely around one person, the robots must also be able to navigate in environments with more people. This can be environments such as pedestrian streets, hospital corridors, train stations or airports. The developed human-aware navigation strategy is enhanced to formulate the problem as planning a minimal cost trajectory through a potential field, defined from the perceived position and motion of persons in the environment. A Rapidly-exploring Random Tree (RRT) algorithm is proposed as a solution to the planning problem, and a new method for selecting the best trajectory in the RRT, according to the cost of traversing a potential field, is presented. The RRT expansion is enhanced to account for the kinodynamic robot constraints by using a robot motion model and a controller to add a reachable vertex to the tree. Instead of executing a whole trajectory, when planned, the algorithm uses a Model Predictive Control (MPC) approach, where only a short segment of the trajectory is executed while a new iteration of the RRT is computed. The planning algorithm is demonstrated in a simulated pedestrian street environment.

One problem with sampling based methods for trajectory planning, is the high number of samples, which are necessary to plan a good trajectory. Therefore, a method, which decrease the computation time for an RRT, such that more vertices can be added in the same amount of time to generate better trajectories, is developed. The algorithm is based on subdividing the configuration space into boxes, where only specific boxes needs to be searched to find the nearest neighbour. The result is an algorithm that can provide better trajectories within a given time period, or alternatively compute trajectories faster. In simulation the algorithm is verified for a simple RRT implementation and in the more specific case where the robot has to plan a path through an environment with moving people.

# Synopsis

Robotter er indtil idag mest brugt som maskiner på fabrikker og som en del af science fiction film. Men indenfor det seneste årti er robotter dukket op i vores daglige omgivelser. I dag findes de for eksempel i form af støvsugere, plæneklippere, legetøjs kæledyr eller som velfærdsteknologier. Hvis vi ønsker, at robotter skal være en endnu større del af vores dagligdag i fremtiden, er det nødvendigt, at robotterne bliver mere intelligente, opfører sig på en sikker og naturlig måde samt tilpasser sig de menneskelige omgivelser. Denne afhandling omhandler det emne; at gøre mobile robotter til en mere naturlig og omgængelig medspiller i de menneskelige omgivelser. Mere specifikt lægges der vægt på sikker og naturlig bevægelse rundt i omgivelserne. Grundtanken er, at robotter skal bevæge sig så menneskeligt som muligt for at blive accepteret af os mennesker i vores hverdag.

I første del af arbejdet er der fokus på at udvikle en robot, som estimerer, om en person er interesseret i interaktion med robotten, dvs. om personen eksempelvis ønsker at tale med robotten. Denne information bruges herefter til at navigere korrekt rundt i forhold til personen. Systemet består af tre subsystemer; et subsystem der estimerer personens position, hastighed og retning i forhold til robotten; et andet subsystem som estimerer om personen ønsker at interagere med robotten; tredje subsystem sørger for at robotten bevæger sig i forhold til personens sociale zoner samt om personen ønsker at interagere. Det udviklede system er herefter modificeret og brugt til at lave et robotspil, som f.eks. kan bruges på plejehjem til at få de ældre til at dyrke mere motion og derved højne deres fysiske helbred.

Robotter i fremtiden skal også kunne navigere gennem et miliø, hvor der er mange mennesker. Det kan være steder såsom gågader, hospitaler, togstationer og lufthavne. Dette ruteplanlægningsproblem er i afhandlingen formuleret som minimering af omkostningen ved gennemkørsel af et potentialefelt. Potentialefeltet er defineret ud fra position og bevægelse af de personer, som opholder sig i området. Dette problem er løst med en Rapidly-exploring Random Tree (RRT) algoritme og en metode til at vælge den mest optimale rute i RRT'en i forhold til omkostningerne. RRT algoritmen er endvidere modificeret til at tage højde for de kinodynamiske begrænsninger, der findes i robottens bevægelse. Dette gøres ved at bruge en 2. ordens dynamisk robotmodel til at udvide træet i algoritmen. Ruteplanlægningsalgoritmen er demonstreret i et simuleret gågade miljø.

Et generelt problem for RRT algoritmer er det store antal samplinger, som skal til, før en god rute kan findes. Derfor er der udviklet en metode, som nedbringer beregningstiden i en RRT. Herved kan der opnås flere samplinger inden for samme tidsperiode, og dette resulterer i at planlægningsalgoritmen kan lave bedre ruter eller alternativt planlægge en tilsvarende rute på kortere tid. Denne algoritme er verificeret ved en simpel generel RRT simulering samt i det ovennævnte gågade miljø.

# Contents

# 1 Introduction

In a not so distant future, we will see robots as a natural part of everyday human environments. The robots will emerge in a wide range of application areas from toys, as household assistants, as customer service robots, to logistic helpers in, for example, hospitals.

For robots to be a natural part of our future daily lives, it is necessary, that they are able to move around safely and naturally in a socially acceptable way. This means, that the motion of the robots have to respect the unwritten social rules that apply, when people move around each other. This Ph.D. thesis focus on the general task of developing algorithms for the safe and natural motion in human environments. To solve this task the thesis incorporate several coherent robot abilities: human pose estimation; estimating human interest in interaction; motion planning according to interest; trajectory planning around many people, as well a optimising planning algorithms.

This chapter first describes the motivation for working with how robots should behave in our future daily lives. Then an overview over the state of the art within Human-Robot Interaction (HRI) and robot navigation research is given. Finally an outline of the rest of the thesis is given.

## 1.1 Motivation

Imagine some time in the future entering the main hall of a huge office building. You have never been there before and need to quickly find a specific room, where an important meeting will start shortly. Around you are a lot of busy business people rushing towards their destination. Intermingling with this crowd is a swarm of robotic agents; small mobile message carrying robots, larger human like robots, flying robots, medium sized wheeled robots, receptionist robots, and not the least visitor assisting robots. In this chaos all the robots must be able to exhibit socially acceptable motion pattern, to be a natural part of the environment, without being in the way. If they are not able to do that, the robots will be an annoyance to the people minding their own business, and they will never be accepted in a human environment. For the visitor assisting robots, it is even more difficult. They will have to identify you as a visitor needing help, and know how to approach you and help you. Even further they will have to agree which robot will help you or to hand you over from e.g. a welcoming robot to a guiding robot. A simplified sketch of this scenario is seen in Figure 1.1, where a service robot leads a visitor to a desired room.

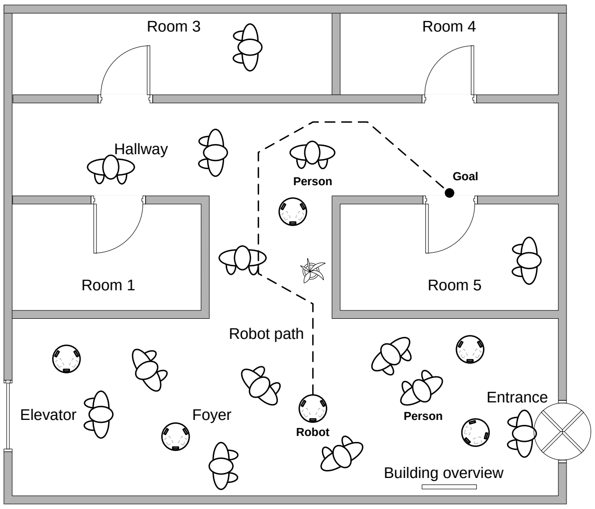
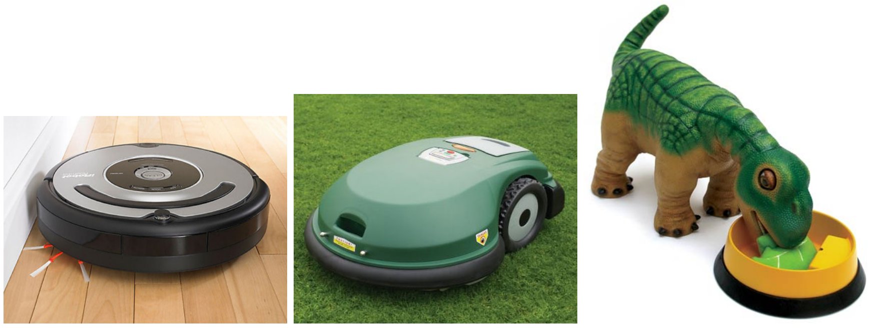


Figure 1.1: Illustration of a scenario, where a service robot helps a person to find a desired room.

This type of urban robots not only have their justification in future office buildings, but in a wide range of environments serving many different assignments. Possible tasks for robots in the future could be:

- assisting people in finding their way in airports;
- assisting people with timetables and directions to platforms at train and bus stations;
- helping people finding the desired goods in supermarkets;
- cleaning your house;
- helping elderly carrying their groceries home;
- delivering packages for postal service companies;
- guiding people and tourists at museums and on city walks;
- transporting medicine, laundry, blood samples etc. at hospitals;
- helping with directing people in large shopping centres;
- functioning as advertising platforms in shopping centres.



(a) Roomba, the vacuum cleaner. (b) A robotic lawn mower. (c) Pleo, an intelligent robot toy dinosaur.

Figure 1.2: Three different robotic devices which are already a part of our everyday lives.

A common, and very fundamental, ability for all those robots is the fact that they need to be able to be close to humans without being in the way. This means that the robots need to move safe and naturally around humans, and in such a way, that it is social acceptable for the humans in the environments.

The scenarios described above are not far from what is seen in science fiction movies. But the fact is, that we have a long way to go before real robots can catch up with their science-fiction counterparts. One of the inventors of the PC-revolution, Bill Gates, predicts that we may be on the verge of a new era, in which future robotic devices will become a nearly ubiquitous part of our day-to-day lives [Gates, 2007]. Robots will enter our daily lives to do floor cleaning, logistics, surveillance, entertainment, rehabilitation training, playing, assisted living, learning and more.

Apart from in movies and in visions for the future, we already see robots entering our domestic environments in increasing numbers each year. Robots that clean your floor, does lawn mowing and plays with your children are already available. Three of this type of robots are seen in Figure 1.2.

So it seems like the robots in our everyday environments have come to stay. However, there is still a long way to go before they are capable of behaving socially acceptable in a complex scenario like the one first described. It is a large challenge for current robotic systems, to be able to act in an unstructured physical environment. One reason is, that the complex environment cannot be completely predefined while the robot is being designed and programmed, but remains unknown until the robot enters the real world. Unstructured physical environments like these, are also called *open-ended environments*.

Besides the obvious comfort of robots assisting in our daily lives, they may also address a more immediate societal problems. One example is, that with the increasing life expectancy, the number of workers compared to the number of retired people, will decrease. There will hence be an increasing need for all sorts of assistance, which can be provided by e.g. robots. For example the number of persons suffering from dementia is expected to more than double before 2040[Dementia, 2010]. A recent study have shown evidence that, for example, regular exercise and cognitive training have positive effect on dementia [Sørensen, 2010]. But if there are no hands, the aging people might not receive proper care. Robots could, for example, help these persons get regular physical and mental exercise by playing games with them, which the care givers does not have time for. The Danish Ethical Committee has recently published a statement "Social Robots", where they foresee the use of social robots for intimate care [Agger, 2009]. So anywhere robots can assist to make more free hands in our everyday lives, it would be of great help.

These visions or ideas of robots as an ubiquitous part of our future lives have, within the last decade, inspired the growth of a new field of research, which is called Human-Robot Interaction. Some of the robots, that have recently emerged in research laboratories around the world are shown Figure 1.3 and Figure 1.4. Most of these robots are designed for special research purposes and have impressive abilities in that respect. But they are not designed for directly fulfilling a task in a human environment. Thus, none of them are yet capable of acting completely autonomous fulfilling assignments in a real-world scenario. Furthermore none of them have a human-aware navigation system that respects social rules of motion. Thus, this is still a missing piece that need to be incorporated in robots for the future.

To substantiate, that there is a potential for robots in our human environment, the Ph.D. project started out with an experiment, where a robot was set at large in a public shopping centre. The robot was named SantaBot, because it was close to Christmas, and it was therefore dressed as Santa Claus. A more detailed description of the experiment follows in the methodology in Section 3.1 on page 41. The experiment is also documented in the paper [Svenstrup et al., 2008]. The SantaBot experiment showed that people are open towards robots, and that there definitely is a great potential for sociable mobile robots in our future environments.

## 1.2 Problem Statement

The SantaBot experiment and the above described future vision for robotics have motivated the further work during this Ph.D. project. Thus, the thesis deals with enabling robots with capabilities to act in future human environments as a safe, natural and sociable actor. More specific, the emphasis is on safe and natural motion and navigation issues. The work in this thesis takes it starting point from the scenario, where a service robot is moving around in an environment like a shopping centre, and airport or an office building. Some of the challenges, which were discovered during the SantaBot experiment, has been further developed into research questions, which are investigated through the thesis.

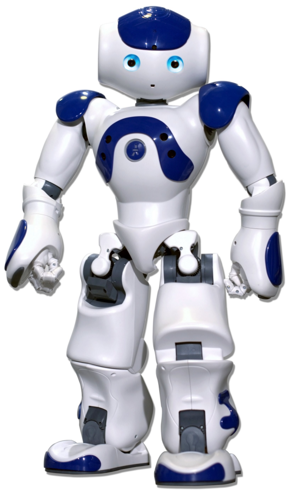
(a) Geminoid is a robot designed by Hiroshi Ishiguro at ATR in Japan.

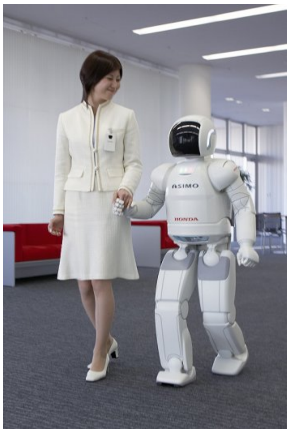
(b) Robovie from ATR, Japan

(c) Paro is a therapeutic robotic baby seal. It is designed to elicit emotional responses to e.g. patients in nursery homes.

(d) Grace from Carnegie Mellon University

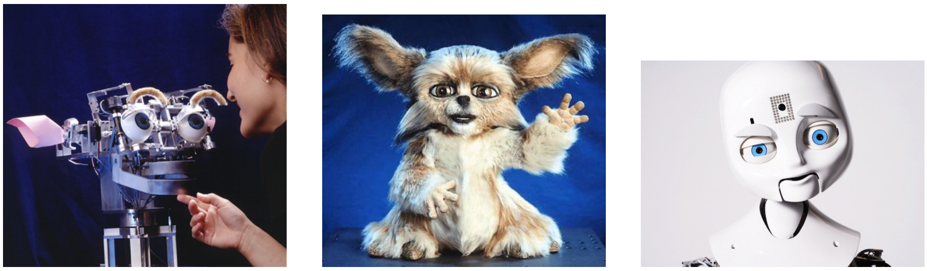
(e) NASA has experimented with Robonaut for HRI purposes.


(f) Nao, the new robot replaced the dog robot AIBO in the standard platform league in the robot soccer world cup.



(g) Honda's robot ASIMO

Figure 1.3: Different robots used for human interaction.


(a) Kismet is capable of being sociable by making appropriate face expressions[Breazeal, 2002]

(b) The successor of Kismet is a fury guy named Leonardo.

(c) Nexi, a new humanoid robot from the MDS class. MDS is short for Mobile/Dexterous/Social.

Figure 1.4: Robots from MIT's Personal Robots Group.

The research questions are:

- Person detection: How can robots detect position and orientation of people by using simple sensors?
- Intention recognition: How can robots find out if people are interested in interaction, based on previous experiences?
- Human-aware navigation: How can a general motion control framework be designed to facilitate sociable and comfortable robot motion around a person?
- Trajectory planning: How can a navigation algorithm be designed to enable robots to navigate through a populated human environment, such that it is safe and natural for the people in the environment?
- Evaluation: How are these abilities evaluated and tested on a prototype robot?

To be able to move around people, it is necessary for the robot to know at least the position and orientation of the people in the environment. Next task, for the robot, is to find out if a person actually needs help and hence wants to interact. Then the robot must navigate naturally towards the person to engage in close interaction, or navigate around the person if the person is not interested in close interaction. The developed intent estimation and motion control framework is shown to be more general applicable for robots with other tasks in a human environment. This is demonstrated by enhancing it, such that the robot is able to play a game with elderly people to motivate them to do a regular amount of exercise. Having solved the question of navigation close to and around one specific person, it is natural to address the situation, where more people are present in the scene. This challenge is illustrated in two images from the real world, which are shown in Figures 1.5 and 1.6. The figures show scenes from pedestrian streets in Aarhus and Tokyo respectively. When encountering environments like this, the question is then, if the robot can safely navigate through the human crowd to a desired end location. Robots, which have been put into the real world, typically handles situations like this by moving relatively slow and relying on people to plan trajectories around the robot and move away from the robot path. Instead, the goal in this work, is to develop an algorithm, that makes the robot move much more dynamic, like the humans in the environment.

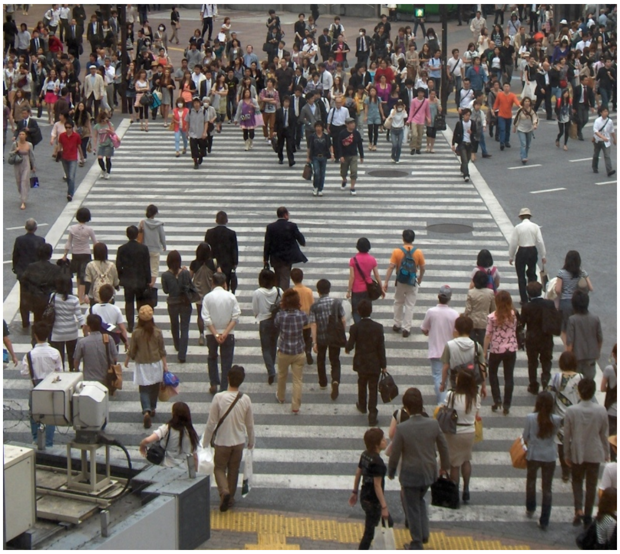


Figure 1.5: An image of a pedestrian crossing in Tokyo.

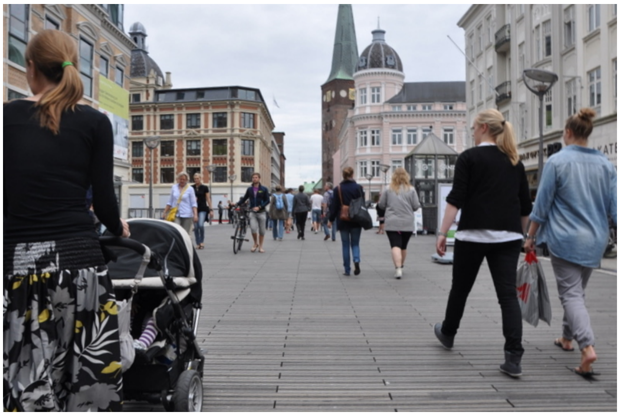


Figure 1.6: A robot perspective of a pedestrian street in Aarhus, Denmark.

## 1.3 Outline of the Thesis

This thesis is written as a plurality of papers, which means that it is divided into two main parts. First an extended summary, consisting of an overview over the topic, current state-of-the-art and an overview over the scientific contributions[1]. The state-of-the-art is presented in the following Chapter 2. Hereafter the methodology in Chapter 3, serves to explain some of the basics methods and algorithms, which are used. In Chapter 4 the highlights of the scientific contributions for each paper are presented, followed by a comparison to other work in 5. Before the included papers a combined conclusion is given in Chapter 6. Second part of the thesis is a collection of the most important publications. This is a brief overview over the subjects treated in the appended papers:

**Paper A [Svenstrup et al., 2009]** Introduces a new Kalman filter based method for pose estimation of persons using simple measurements from a laser range scanner. Being able to estimate both position and orientation of persons is very important for being able to navigate safe and naturally in a human environment. Furthermore the development of a human-aware navigation algorithm and human intention estimation, which is more thoroughly described in *Paper B*, is shortly introduced as well.

**Paper B [Svenstrup et al., 2010b]** Presents a combined system, which is able to detect human pose, estimate human interest in close interaction with the robot, and adapt the motion accordingly. The pose estimation is used in a Case-Based Reasoning (CBR) system, that determine if persons are interested (or not interested) in interacting with the robot. Based on the estimated interest in interaction, a human-aware navigation algorithm ensures that the robot moves appropriately relative to the person. Experiments demonstrate that the combined algorithms work on a real robotic system.

**Paper C [Hansen et al., 2010]** Demonstrates that the framework described in the first two papers is general enough to be used in other applications for robots in human environments. Specifically in this paper, the framework is further developed into a robot pursuit evasion game, where a player tries to get close to the robot. The game is intended to motivate elderly persons to do a regular amount of physical exercise, which in the long term can improve their health. Instead of estimating the interest in interaction, the robot in this paper, estimates the skill of person. If it is a low skilled player, the robot approaches the player to make the game easier, and if it is a player with good skills, the robot tries to avoid the player, according to the motion scheme developed in *Paper A and B*.

[1] http://phd.ins.aau.dk/intranet/templates/3863476

**Paper D [Svenstrup et al., 2010a]** The human-aware motion planner from *Paper A and B* is extended to a trajectory planning algorithm, which is able to plan a trajectory through dense human crowds, like experienced in for example pedestrian streets or airports. First the problem is formulated as a minimisation problem. Then a trajectory planner, based on a modified Rapidly-exploring Random Tree (RRT) algorithm, is designed to find the solution to the minimisation problem. The RRT algorithm is modified to take into account real-time operation and the dynamics of the moving robot. In simulation it is shown that the algorithm ensures safe robotic motion through a pedestrian street.

**Paper E [Svenstrup et al., 2011]** Presents a new method to minimise the computational complexity of an RRT. Minimising the computational complexity of an RRT ultimately enables the algorithm to make faster or better plans. It is shown, that the algorithm presented in *Paper D*, is able to generate better trajectories using the new method.

### 1.3.1 Additional Scientific Communication

In addition to the publications, which are included in the thesis, the following publications are associated with the work on the Ph.D. project as well:

1. Svenstrup, M. (2011). Navigation among humans. In *Advances in Robot Navigation*, editor Barrera, A.. INTECH, June 2011. ISBN 978-953-307-1345-8

2. Svenstrup, M. (2010a). Sampling based trajectory planning for robots in dynamic human environments. In *Extended abstract in workshop material from Robotics Science and Systems (RSS) 2010.*

3. Svenstrup, M. (2010b). Sampling based trajectory planning for robots in dynamic human environments. Poster at workshop Motion Planning: From Theory to Practice at Robotics Science and Systems (RSS) 2010.

4. Svenstrup, M., Bak, T., Maler, O., Andersen, H. J., and Jensen, O. B. (2008a). Pilot study of person robot interaction in a public transit space. In *Proceedings of the International Conference on Research and Education in Robotics - EUROBOT 2008*, Lecture Notes in Computer Science, pages 120–131, Heidelberg, Germany. Springer-Verlag GmbH.

5. Svenstrup, M., Andersen, H. J., and Bak, T. (2008b). Adaptive robot to person encounter. Poster at the 3rd ACM/IEEE international conference on Human-Robot Interaction (HRI).

6. Hansen, S. T., Svenstrup, M., Andersen, H. J., and Bak, T. (2009a). Adaptive human aware navigation based on motion pattern analysis. In *Robot and Human*

*Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages –, Toyama, Japan.

7. Hansen, S. T., Svenstrup, M., Andersen, H. J., Bak, T., and Jensen, O. B. (2009b). The santabot experiment: a pilot study of human-robot interaction. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 211–212, New York, NY, USA. ACM.

8. Hansen, S. T., Svenstrup, M., Andersen, H. J., Bak, T., and Jensen, O. B. (2009c). The santabot experiment: a pilot study of human-robot interaction. Video for the 4th ACM/IEEE international conference on Human robot interaction (HRI) video session.

9. Andersen, H. J., Bak, T., and Svenstrup, M. (2008). Adaptive robot to person encounter. In *Proceedings of the International Conference on Research and Education in Robotics - EUROBOT 2008*, Lecture Notes in Computer Science, pages 13–23, Heidelberg, Germany. MATFYZPRESS.

In addition to the above publications, the work has also inspired more public domain scientific communication. For example a couple newspaper articles (in danish):
http://jp.dk/nyviden/article1478023.ece
http://videnskab.dk/teknologi/robot-laerer-af-menneskers-bevaegelser

Additionally the robot, which is described more in detail in Section 3.5 on page 51, have also participated in exhibitions and other various events to promote robotic research and university educations in particular.

# 2 State of the Art and Background

This chapter provides the background and current state of the art related to the different subjects treated in this thesis. The subjects, which are outlined in Section 1.2, can be categorised into two different research areas. The person detection, the intention recognition and the human-aware navigation subjects are directly linked to HRI research. Therefore a brief and general overview over HRI is given in Section 2.1, and then more detailed state-of-the-art descriptions of the subjects are presented. Second research area is more generally about robot navigation, trajectory generation and motion planning. In Section 2.2 an overview over research in this area, and a more thorough analysis of the methods related to the thesis, are given. Each subsection gives an overview over the state-of-the-art research and concludes with an evaluation of how the work in this thesis fits in.

## 2.1 Human-Robot Interaction (HRI)

HRI lies in the intersection of robotics, artificial intelligence social sciences and psychology [Dautenhahn, 2007](see Figure 2.1). To be able to interact with humans, the robots need basic robotic skills; they need intelligence; and they need to behave in a socially acceptable way. The HRI research area started growing roughly a decade ago, and has been a fast growing ever since. A highlight was in 2006, where the *first annual conference on Human-Robot Interaction* emerged. Recently some large European projects [COGNIRON, 2007, COSY, 2007] concerning HRI have been running, and more is likely to come. [Fong et al., 2003] surveys socially interactive robots, i.e. the current status as of 2003. Although it is some years old, it presents a good and still relevant overview of basic research issues and themes within socially interactive robots. Fong has a point, which is very relevant in the way we approach HRI:

> *"Humans prefer to interact with machines the same way that they interact with other people."*

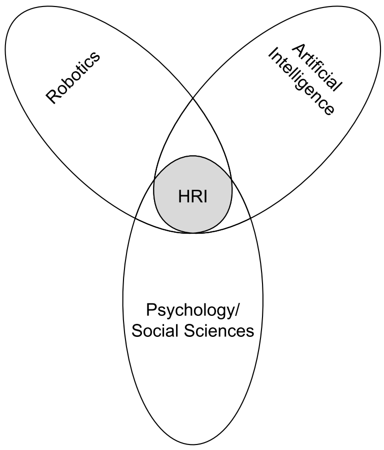


Figure 2.1: HRI is at the intersection of Robotics, Artificial Intelligence and Psychology/Social Sciences.

This statement is central for the ideas behind the developed algorithms in this thesis. This means that for robots to get accepted in human environments, they must act in a similar way as humans would do.

A relative thorough and more recent survey of HRI is presented in [Goodrich and Schultz, 2007]. Goodrich splits the HRI research into two distinct categories; remote interaction and proximate interaction. Remote interaction is where the robots and humans are physically separated like for example the interaction software that controls the Mars Rovers. The proximate interaction is where the robots and humans are at the same physical location, which is what is interesting with respect to this thesis, where the robots enter the human spaces.

There are quite a few distinct subjects areas within proximate HRI. Most of the research within proximate HRI can be categorised within one or more of the subjects illustrated in Figure 2.2. The grouping into these areas is based on an analysis of HRI related publications at conferences like HRI, RO-MAN, IROS and ICRA. The subjects marked with green are the ones, which are directly treated in this thesis. The ones marked with blue have some background relation to the work, which means that the areas are not explicitly treated in the thesis, but have some relevance for the subject of the thesis. The white ones are not directly related the work presented in the thesis. The subjects, that are not directly treated in this thesis, are shortly presented here:

- **Facial expressions:** Robots that express human emotions, to be able to be sociable (see e.g. [Breazeal, 2002]).
- **Roboethics:** As we see more robots in our natural environments, a discussion on how they should be allowed to behave is very relevant. Movies like; *Blade Runner*;

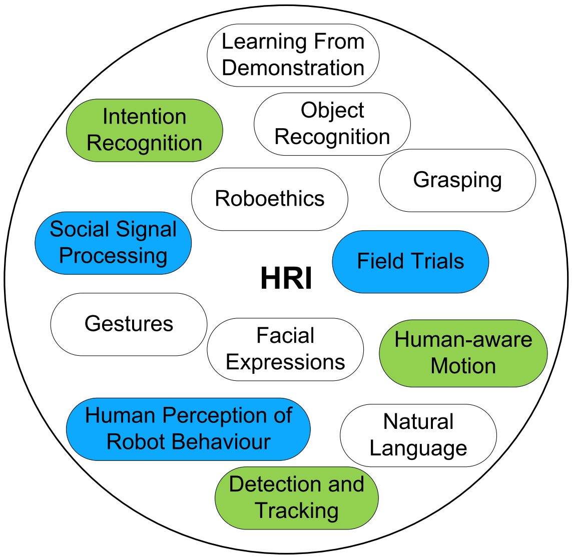


Figure 2.2: Research within HRI can be grouped into a range of areas. The green areas are the ones that are closest related to the work presented in this thesis. The blue areas have relation to the thesis, and the white ones are not directly coupled to the thesis.

*Terminator*; *A.I.*; *I, Robot*, etc., contributes to the popular belief that robots might be dangerous when becoming intelligent parts of our everyday environments (see e.g. [Asimov, 1968, Arkin, 2008]).

- **Natural language:** Robots that communicate verbally with humans, both talking and understanding the language. In fact, because natural language is difficult, simpler methods for conveying information, like text on a screen, is also a research issue. Natural language processing has been studied extensively for many years in computer science for human-computer interaction (see e.g. [Brick and Scheutz, 2007, Manning and Schütze, 1999]).
- **Grasping:** To move things in the physical world, robots need to be able to grasp the things (see e.g. [Pratichizzo et al., 2010]).
- **Learning from demonstration:** Humans learn many of their skills by watching and mimicking other people, thus robots that learn to perform tasks by having a teacher showing it, have a great potential (see e.g. [Calinon and Billard, 2007])
- **Object recognition:** If you say "please hand me the red apple", the robot must be able to recognise the apple. Computer vision is usually used for these tasks (see e.g. [Stiefelhagen et al., 2007])
- **Gestures:** This area concerns understanding human non-verbal gestures like pointing, waving, etc. (see e.g. [Jenkins et al., 2007]). The area is often also denoted

pose estimation, which is not to be confused with the definition of pose used in this thesis (see Appendix I on page 208).

- **Social signal processing (SSP):** Research that addresses the ability to recognise human social signals and behaviours [Vinciarelli et al., 2009]. SSP is not a focus in this thesis, but it has some resemblance to what is done, since we estimate human intentions by observing the behaviour of the person.
- **Human perception of robots:** How does people actually feel, when a robot enters their natural environments (see e.g. [Giusti and Marti, 2006, Kulic and Croft, 2007, Koay et al., 2005, Sverre Syrdal et al., 2006]). This is also relevant in relation to this Ph.D. project, where it is desired that the robot moves naturally for the people in the environment. But no experiments, where the human perception is measured, is done. However, it is relevant in relation to how a robot should move, and is thus described further in Section 2.1.3.
- **Field trials:** This area concerns experiments, where robots have been put out into the real world to find out about their potential (see e.g. [Shiomi et al., 2008], [Mutlu and Forlizzi, 2008],[Kanda et al., 2007]). The SantaBot experiment (see Section 3.1) is a field trial experiment, and thus, the area has some connection to this thesis. Many of the works described below also contains field trials to see how the robot behaves in the real world, or a controlled real world setting.

Based on the scope of this thesis and the research questions presented in Section 1.2, the three central themes that are relevant in relation to HRI are; human detection and tracking, intention recognition, and human-aware motion. These are described more in detail in the following sections.

### 2.1.1 Detection and Tracking of People

This section describes different approaches to human detection and tracking. There are several different types of sensors and combinations thereof, which have been successfully used for people tracking. This is for example 2D and 3D vision [Dornaika and Raducanu, 2008, Munoz-Salinas et al., 2005, Jenkins et al., 2007] and thermal tracking [Cielniak et al., 2005]. But in applications, where it is only desired to find the persons position and not posture or face expressions, laser scanner based people tracking is prevalent (see e.g. [Fod et al., 2002, Kluge et al., 2001, Xavier et al., 2005, Gockley et al., 2007]). Laser scanner data have also been used in combination with vision systems, where the laser data is used to find the person, and the vision systems are used to obtain further information, like pose [Feil-Seifer and Matarić, 2005], [Michalowski and Simmons, 2006, Bellotto and Hu, 2009]. The success of the laser based methods is because it is relative easy to extract information about a person's legs from a simple line scan compared more sophisticated image processing. Furthermore the laser based approaches tend to work at longer ranges than vision. But the different types of sensors have different capabilities. For example, using vision enables more accurate tracking in terms of finding features like the posture of the person. Also face tracking [Sisbot et al., 2006, Pateraki et al., 2009, Kleinehagenbrock et al., 2002], is important for close interaction purposes, where it is necessary to find out if a person is looking at the robot, and what kind of facial expression the person has.

There are basically two categories of leg detection algorithms. One type finds the motion between each scan and detects persons from that, using, for example, particle filters. In several cases an iterative closest point scan matching technique is used to match consecutive scans and thus being able to find the motion. This type of method relies on the person to move, otherwise the person will be treated as a static obstacle. Examples of research papers that applies this type of algorithm is [Lu and Milios, 1994, Montemerlo et al., 2002, Gockley et al., 2007, Rodgers et al., 2006, Zender et al., 2007, Kondaxakis et al., 2009].

In the second method, geometric feature analysis of legs are used to find the legs or hypotheses for legs being present. The laser range data are analysed for leg sized convex patterns, either one of them or two at a reasonable distance from each other. In other words, the characteristic rounding of a leg is a feature in the laser scan. Another feature is, that most people have two legs, which gives an easy recognisable pattern in the scan. Papers that have successfully implemented systems belonging to this category is, for example [Kleinehagenbrock et al., 2002, Schulz et al., 2001, Kluge et al., 2001], [Topp and Christensen, 2005, Arras et al., 2007, Xavier et al., 2005].

An inherent problem arising from detecting persons from a robotic platform, is that the robot moves. Especially using image processing this is a problem, since the same fixed object moves and transforms from frame to frame. This is easier accounted for in laser based tracking, where a higher sampling frequency makes the changes smaller from sample to sample. Furthermore, if the robot motion is known, it is relatively easy to compensate for the motion in a one dimensional range scan using e.g. a particle filter like in [Gockley et al., 2007].

There also exist a third method for detecting people. This method is a bit different in that it is based on multiple stationary laser range finders that covers the environment from different positions. In this case it is possible to do background subtraction, since the background does not move, which makes it easier to detect moving obstacles. In [Zhao and Shibasaki, 2005] the laser range finders are mounted in about $20cm$ height, in [Fod et al., 2002] it is about waist height ($0.8m$) and in [Glas et al., 2007] the lasers are mounted in about torso height of the humans. Common to these algorithms is that they first gather and combine the measurements from all the sensors and after that applies an algorithm to find the people in the area. These systems works well, but the downside is that the sensors are not mounted on the robot, and thus, the obtained person positions are not relative to the robot. Furthermore a robot would be dependent on the surroundings for information, since the laser is not an on-board sensor.

**Summary of detection and tracking of people**

The goal in this Ph.D. project, is to enable natural motion of a robot in a human environment. For this to be possible, an algorithm, which can detect persons at a long range, is necessary. To facilitate more sociable motion around a person, in addition to the position of a person it is also necessary to know the orientation of the person. Furthermore the algorithm must be computationally light, since it has to work on-line on a robotic system, where a other algorithms are running at the same time. Vision based methods do not work well on long distances, mainly because of resolution and robot motion reasons and they often also take much computational power. Laser based person detection algorithms are relative easy to implement and are typically computationally light. There are, as described above, many successful approaches to detection and tracking of humans using laser range scanners. Thus, for the purpose of this project, we do not want to develop a tracking algorithm from scratch. It has been chosen to apply the algorithm described in [Xavier et al., 2005] as a starting point.

However, one drawback of all the laser based algorithms described, is that they only provide the position of the person, and no other information. This might be good enough for person tracking, but not for sociable robot motion. Therefore it is necessary to extend on the laser based methods to also obtain an orientation estimate.

### 2.1.2 Recognising Human Intentions

Robots in human environments face many different challenges and situations, which can be difficult to predict when programming a robot. For example, to understand human intentions from observations is a hard task, and since it cannot be pre-programmed, the robots need to be able to learn to anticipate how people will behave. There exist much literature for machine learning in general (see e.g. [Alpaydin, 2004]), but estimating human intentions has not been widely studied within the HRI community. Most of what is done in relation to learning in HRI either uses a long training period, or is closely related to learning by demonstration, where a robot have to learn to mimic human actions. However, estimating intentions cannot be demonstrated, and is thus a different problem.

There are, however, a few examples where intention estimation has been tried. For example, in [Schmidt-Rohr et al., 2008], a robot tries to estimate the intentions of a person in terms of *what does the person want me to do?* Here a Partially Observable Markov Decision Process (POMDP) approach is used to reason based on the sensory input, and act according to the belief of the intention. However, the reasoning becomes very complex because of different input (vision and speech), and many possible tasks, and is thus not very intuitive. A simpler approach is used in [Kelley et al., 2008], where person movement relative to each other is used to estimate the intention of the persons. The underlying method used for estimating the intentions is Hidden Markov Models (HMM). In experiments a robot is used to find out if a person wants to: follow another person, meet with another person, pass the person, drop of things, or pick up things. The algorithm performs well, however it is based on a vision system and is only demonstrated to work on a stationary robot. In [Feil-Seifer and Matarić, 2005] joint intention theory (see e.g. [Smith et al., 1998]) is used to find out if people are interested in interaction. They use vision, laser range scanner and speech recognition to sense the persons. The goal of the robot is to determine which person is interested in interaction. The success rate was $82\%$ at close range, and $62\%$ above $3m$. Bayesian inference algorithms and Hidden Markov Models have also successfully been applied to modelling and for predicting spatial user information in [Govea, 2007].

Common to these methods are, that they are pre-trained, and thus not able to adapt to new situations, i.e. the methods do not forget what they learned yesterday if they start to see new behaviour patterns, which means, that they are not able to learn new behaviours on-line. This is important, since a robot in a future world needs to be able to adapt to a changing environment, which cannot be completely defined already when programming the robot before it is deployed in the real world. Furthermore, it is wanted to have an algorithm that is more intuitive perceivable, i.e. reasons in a way that resembles the way a human would do, and can handle arbitrary sensory input that different robotic systems can provide. A method that is able to do this, is Case-Based Reasoning (CBR). CBR fulfills the above described requirements, but has so far not been used for intention estimation purposes. CBR allows recalling and interpreting past experiences, as well as generating new cases to represent knowledge from new experiences. CBR has proved successful in solving spatial-temporal problems in robotics in [Kolodner, 1993, Likhachev and Arkin, 2001, Ram et al., 1997, Jurisica and Glasgow, 1995]. A deeper description of CBR is found in the methodology (see Section 3.3 on page 45).

### Human motion prediction

Recognising peoples intentions is not only a question about what people intent to do in relation to the robot, but also where they intent to go, i.e. the intended trajectory. Estimating and predicting peoples trajectories can be of great help for planning motion around humans. A simple model for the human motion when moving towards a goal is presented in [Fajen and Warren, 2003, Fajen et al., 2003]. The model is derived from physical experiments with people given a goal to walk towards. From the results the authors suggests that: “human route selection does not require explicit planning, but may emerge on-line as a consequence of elementary behaviors for steering and obstacle avoidance". For a constant velocity, goals function as simple attractors for the heading and the obstacles functions as simple repellors for the heading. The steering (rotational speed) of the person is then calculated as a simple proportional controller:

$$\dot{\phi} = -k_g(\phi - \psi_g) + k_o(\phi - \psi_o)\exp(-|\phi - \psi_o|) \tag{2.1}$$

where $\phi$ is the persons direction, $\psi_g$ is the direction of a goal, $\psi_o$ is the direction of an obstacle, and $k_g, k_o$ are constants. An illustration of these angles is shown in Figure 2.3. While this motion model is interesting for modelling of human motion, it is still difficult to predict the motion of a human, if the goal is not known.

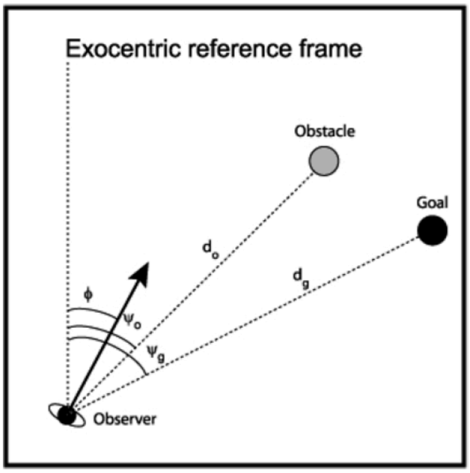


Figure 2.3: The angles from Eq. (2.1). The figure is shown in [Fajen et al., 2003].

Other approaches tries to find characteristics or features of different trajectories in a given environment. Motion of people in the environment are then compared to these features and a predicted trajectory can be extracted from the most probable trajectory. Since the robot has a better prediction of where persons are in the future, it will have a higher probability of generating a path that do not interfere.

For example, in [Foka, 2005, Foka and Trahanias, 2010] manually defined or learned so called hot points are used to predict, where people may go. Hot points may be where you would usually find people, like chairs, doors, stairs, exhibits in a museum, ticket counters, entrances etc.. Depending on the direction of people in the area the probability of each hot point being a goal, is calculated. The probability of each hot point being a goal is then integrated into a POMDP motion planning algorithm for the robot (see Section 2.1.3). The hot points can also be learned, where each point in the map is assigned a probability for being a hot point.

In [Bennewitz et al., 2005, Bennewitz, 2004], a method using HMM for learning collections of trajectories that characterise typical motion patterns of people in an environment. The approach uses the Expectation-Maximisation algorithm to cluster trajectories recorded with laser-range sensors into a collection of motion patterns, which are applicable for the given environment. Prediction is done by automatically deriving a HMM based on the typical motion patterns. When an obstacle moves, the possible set of motion trajectories converges to a single class, and can thus be used in the planning process for the robot. Another similar method, which is also based on that the human motion is goal oriented towards a finite number of points of interest, is proposed in [Bruce and Gordon, 2004]. Here the information is used to improve tracking accuracy by predicting where people will go, when they for example get occluded by obstacles or other people. Initially, a prerecorded laser data set of point estimates of peoples positions, are used to find trajectories. The trajectories are then clustered by hand (but the authors suggest that it can be done like in [Bennewitz et al., 2002]) to find maximum likelihood of goal locations. During tracking of a person, a planner plans paths from an observed person's last observed position to the goals. The resulting paths are used in the tracker to predict where the person might go, and thus increasing the accuracy of the tracker. An additional note is that this motion prediction of the people could be used in a path planning algorithm for the robot, which is not done in the paper.

The above described methods for estimation human trajectories all assumes some preliminary knowledge of a map, potential goals or sampled trajectories of people. This is of cause a valid assumption, when a robot is moving around in the same environment all the time, and can be used for museum tour guide robots, service robots for specific areas, domestic robots etc. However, if we want more generic robots, that can handle all sorts of tasks, like e.g. a personal assistant for elderly or an autonomous wheelchair, we cannot assume that such characteristics of the environment is known in advance.

**Summary of recognising intentions**

There have not been much work on recognising human intentions for interaction purposes. We have chosen to base the development of a system that estimates a person's interest in interaction on a CBR system. CBR is easy perceivable and has already proved successful in solving spatial-temporal problems in robotics. There have been some work on learning to predict human motion in an environment, but these works have not taken possible interaction into account, and are thus not investigated further in this thesis.

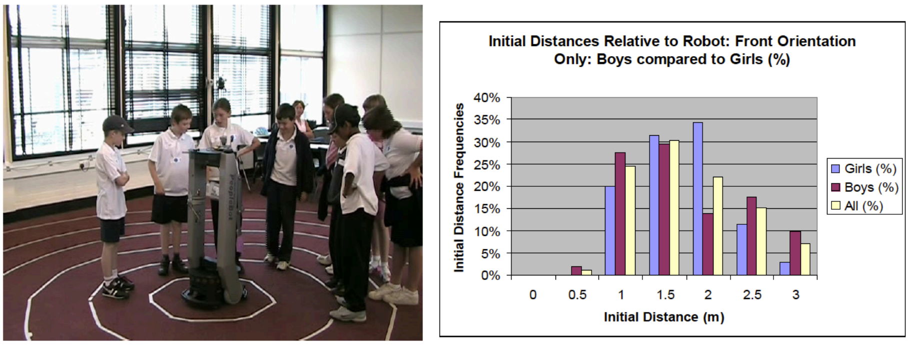
### 2.1.3 Human-aware Motion

The aim of the work in this thesis is centered around enabling a robot to move, such that it is socially acceptable to people. That means robots have to move naturally, comfortable and safely for people in the environment. Because of this focus, this section is the most relevant part of the state of the art. Thus, an in depth description of the different systems is therefore given in this section.

Regarding robot motion in human presence, there are mainly two research issues that are treated in literature: how should a robot move, i.e. how persons feel that the robot should behave; and motion algorithms that are actually implemented. To the first category, the literature mostly contain user studies, where test subjects are asked how they feel for different behaviours of the robot. For the implemented motion schemes, they are mostly focusing on avoiding collisions with people.

### How should a robot move

To find out how a robot should move in a human environment, a good place to start, is to study how humans position themselves relative to each other. Already in 1963 Edward T. Hall published a large study of human-human spatial placement behavior is [Hall, 1963, Hall, 1966]. He called his theory “proxemics theory”, where he categorise distances between people into four classes. The distances are called, intimate, personal, social and public. The four zones governs with type of interaction happens at the given distances. This theory is used in the thesis as a basis for a mathematical formulation of where the robot should place itself. Therefore a further description of the proxemic theory is given in Section 3.2. These spatial zones are static zones, which apply to how people position themselves relative to each other. But if we turn to moving robots, [Yoda and Shiota, 1995] made a study on how humans pass each other, which is used to generate robot motion in [Yoda and Shiota, 1997]. In [Pacchierotti et al., 2006a] a passing a person user study, where people in a corridor evaluated the behaviour of a robot that acted according to the proxemic theory, were performed. Results are that the preferences for robot motion comply with the proxemic theory. Furthermore ideal values for robot speed and lateral distance for passing in a hallway was found (see also Figure 2.7). It turned out that both large avoidance maneuvers or too close passings felt uncomfortable or unnatural. Another study, [Walters et al., 2005a], considers close encounters. Participants (both kids and adults) were asked to approach a robot, and stop at a distance, where they felt comfortable. A photo from the experiment with children is shown in Figure 2.4, together with a histogram over the experimental data with the comfortable stopping distances. As can be seen in Figure 2.4a, the children mostly place themselves in the social zone of the robot, but on the border to the personal zone ($1.2m$). The result for adults was that the approach a bit closer to the robot. In the second part of the study, the roles were inverted, such that the robot approached the persons. In this experiment, the robot was allowed to approach a bit closer, than the participants approached in the first part of the experiment. Furthermore the paper conclude that there is a need for long term trials, since social distances may change over time. A



(a) Children position themselves, where they feel comfortable relative to the robot.

(b) A histogram over the comfortable stopping distances.

Figure 2.4: Two figures from [Walters et al., 2005a], that study how people position themselves relative to the robot.

long term field is for example performed in [Kanda et al., 2007], where a robot stayed in an elementary school for two months. However, this study is mainly limited to finding out how children change attitude and establish friendly relationships to the robot over time, and not so much spatial behaviour. Follow-up studies on the stopping distances were performed in [Koay et al., 2005, Dautenhahn et al., 2006, Sverre Syrdal et al., 2006, Koay et al., 2007]. Here, various experiments are done with different approach directions; a comfort level device measuring peoples comfort; and approaching when handling and handing over objects. One interesting result is that it turns out that humans prefer to be approached from a $45°$ degree angle.

In the above described experiments, the robot was standing still, or remotely controlled. So the next small step to introducing mobile robots into human environments, is to make simple experiments, where a robot moves autonomously around in a human environment. One type of experiments it to make robots simply follow a person around. [Gockley et al., 2007] is describing natural person following behaviour, in terms of being a social robot. Two ways of person following are designed; a direct path following algorithm, and a direction following algorithm, where the robot always moves directly towards the person. The direction following approach implies that the robot will cut corners, if, for example, the person is turning. A user study was conducted, where people observed the robot following a person, and the users had to evaluate which following behaviour seemed more natural. The robot tried to stay at a distance of $1.2m$ from the person. The result was that the direction following algorithm was by far the most natural, and the users tend to think that the robot was a little too far away.

Other, more recent studies also include user studies on proxemics. For example on gaze direction, i.e. how social spaces change depending on if the person's head are turned towards the robot. It is also studied how the social spaces depend on the person's sex, the experience with robots, the person's personality traits etc. [Takayama and Pantofaru, 2009, Walters et al., 2005b, Mumm and Mutlu, 2011]. Other experiments in [Nabe et al., 2006, Satake et al., 2009, Yamaoka et al., 2008] measures human behaviour in relation to the robot and how to approach a human and how close to approach. However these experiments present small changes in the personal spaces based on very subtle changes in person's behaviour and personalities. Therefore it is, although relevant, difficult to implement in a robotic system, since it requires sensors that are able to sense these subtle signals.

As described, there have been some research confirming the validity of the proxemic theory for robots as well as for humans. Furthermore there are additional results on more specific robot motion. For the case of this project, we do not perform any additional user studies, but rely on the above presented works. Especially the social zones from [Hall, 1963], and the reported preferred approach direction from [Dautenhahn et al., 2006].

### Implemented motion schemes

One of the first attempts for defining a motion planning algorithm for a robot that coexists with humans is found in [Tadokoro et al., 1995]. Here, a framework for motion in the human environment is presented. Sensors perceive the persons and a stochastic motion predictor, is used to find out where the people go. This information is used in a genetic algorithm that plans a path and executes on the robot. However, no experiments with the robot moving in a human environment were done. Initial attempts to put a robot in a real human environment includes [Nakauchi and Simmons, 2000]. Here it is attempted to apply the social rules of standing in a line to a robot named Xavier. The personal spaces of people standing in line have experimentally been measured. The result is seen in Figure 2.5, where it can be seen that the personal space is larger towards the front, than towards the back of a person. In an experiment, the robot was able to detect the line, enter it and move up to the head of it in 70% of the trials. Although [Nakauchi and Simmons, 2000] defines a personal space, it can not be used for general robot navigation, since it is measured only for people standing in line. However, if implementing a robotic system, which should be able to also stand in line, it is necessary to take this personal space into account.

**Person following:** As described above, a natural and relatively simple first step to making robots move around in human environment, is to design robots that follow a person. Thus a number of researchers have worked on this issue.

[Yoshimi et al., 2006] has developed a person following robot named ApriAttenda$^{\text{TM}}$. The robot is equipped with cameras that detects and track persons around the robot. Spe-

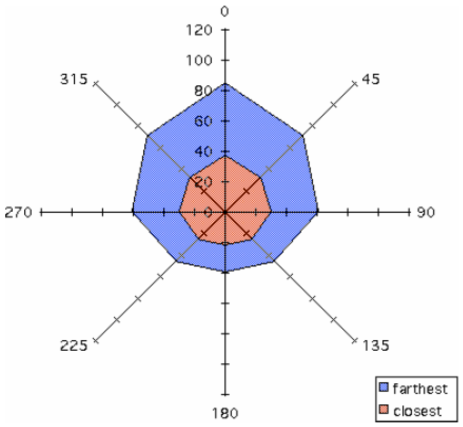


Figure 2.5: The experimentally determined personal space of persons standing in line from [Nakauchi and Simmons, 2000].

cific persons is recognised by their colour and texture of the clothes. The robot uses a very basic motion behaviour, where it tries to follow a person at a distance of $1.5m$, while avoiding known obstacles. The robot tries to always face directly towards the person, and thus the tracking algorithm is similar to the most natural behaviour, as described in [Gockley et al., 2007] above. Other work related to the person following, is done in [Prassler et al., 2002]. Here the focus is to make the robot navigate safely side-by-side with a person. The test scenario is a wheelchair navigating beside a person. In this case the method known as Velocity Obstacles (see Section 2.2.2) was used to avoid collision with other people. In the experiment they moved around at a railway station for a total distance of $3km$ over $8-10$ hours without collisions. In other autonomous and semi-autonomous wheelchair applications are presented in [Argyros et al., 2002, Gulati and Kuipers, 2008]. [Argyros et al., 2002] describes a semi-autonomous wheelchair that is able to follow a target, move in the middle of a free space and move in a desired direction. [Gulati and Kuipers, 2008] introduces "graceful" motion for a wheelchair, which is defined as being safe, comfortable, fast and intuitive. Even though humans in the environment are not taken into account, the criteria for graceful motion was satisfied. This is however more on a theoretical motion planning level and not really considering humans in the planning.

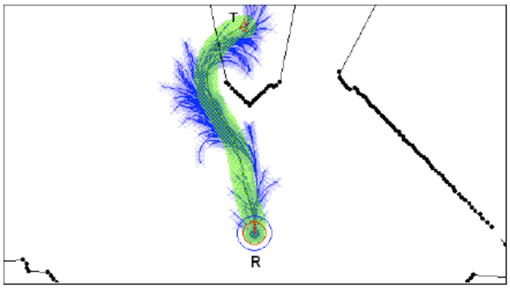
In [Takemura et al., 2007] a potential field based navigation strategy is applied to follow a person and to avoid obstacles as well. The person is assigned an attractor potential and obstacles are modelled as repulsive forces. The forces coming from the potential are calculated as the reciprocal of the distances to the objects. The robot then moves in the steepest decent direction to approach the goal, while still avoiding obstacles. The problem with potentials, is that the robot can be trapped in a local minima, if the robot e.g. is facing directly towards an obstacle (see Section 2.2.2). This is solved by increasing the potential at the local minima.

In [Zender et al., 2007] the person following algorithm is enhanced to track and follow persons in a socially acceptable way. This is done by using simultaneous localisation and mapping (SLAM) to understand where the robot is, in the environment, and what person action to expect. However, it is not explicitly discussed how the term socially acceptable motion is defined. Only, that the robot can adjust its behaviour to obtain a more socially acceptable motion pattern, according to the knowledge of the position in the environment. The robot uses verbal and non-verbal feedback to the user where necessary. One way the robot exhibits situational awareness, is when the user approaches a door. Under normal circumstances the robot is following at a distance of $0.5m$, but near the door it is increased to $2m$. An other form of situational awareness, is when moving in a corridor, where the robot's maximum allowed speed is increased. This algorithm needs a map of the environment with specified property labels, like door, corridor etc..

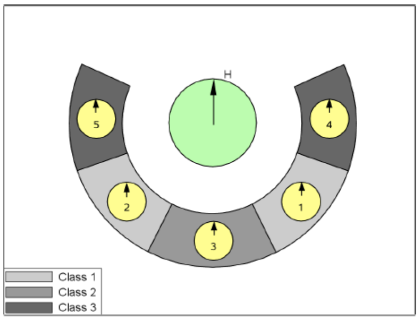
**Accompanying persons:** An opposite scenario of a robot following a person, is a robot guiding a person. This is an inherently easier task, since it is the robot, who determines where to go. However, if a group of people has to be guided, the robot needs to make sure that all follows correctly. In [Martinez-Garcia et al., 2005], a scenario, where a group of robots guide a group of people. The guiding takes place with no explicit communication, and only motion of the robots implicitly guides the group.

A slightly different situation is accompanying one person. This issue is treated in [Hoeller et al., 2007]. The paper presents a local navigation planning approach for avoiding collisions, while simultaneously staying close to the client. The core of the local navigation is similar to the Dynamic Window Approach (DWA), which is described in Section 2.2.2. The robot plans an extensive space tree from the current position using a robot motion model with the state vector $[x, y, \theta, v, w]_k^T$, where $v_k$ and $w_k$ are velocity and rotational speed input. The model assumes that the velocity changes are carried out instantaneously. The tree is generated by a sampling based approach, and can be seen in Figure 2.6a. The robot guesses a short time target for the person that is accompanied. This target is the goal for the expansive tree. After generating the tree the optimal path, highlighted with green in Figure 2.6a, is extracted. Additionally the area behind the human is divided into five zones, as shown in Figure 2.6b, where the robot chooses the zone, which have the highest priority (the numbers on the figure) and not being blocked by obstacles.

Now, moving to a bit more complex scenario, [Müller et al., 2008] presents a path planning algorithm that enables the robots to move efficiently and smoothly with groups of people. The goal is to generate a trajectory that makes the robot move similar to humans. The robot selects those individuals that move towards the robot's desired goal and follows them. The A* planning algorithm is iteratively used to find the optimal path.



(a) An expansive tree of possible robot trajectories.



(b) Five potential positions for the robot behind the human, which is being followed.

Figure 2.6: An expansive tree and the different zones as illustrated in [Hoeller et al., 2007].

The environment is defined by two occupancy grid maps. A global static map, and a local map containing the observed dynamic obstacles. The A* algorithm operates on the combination of the maps, while the heuristic used in the algorithm only works on the static map. To follow people going towards the same direction as the robot, the grid points they are occupying are marked as free, which makes the algorithm plan a path similar to theirs.

**Autonomous motion:** If we want robots to be a natural part of our environment, they must not only be able to follow and accompany people. They must also be able to move by themselves, and at speeds equal to humans. If a robot has to operate at higher speeds, their dynamics must also be taken into account. In [Shi et al., 2008], some initial results on a human-aware motion planner for a Segway RMP platform is presented. An RRT (See Section 2.2.1) based planner is used to plan a path in the environment. Additionally some simple user response experiments are performed. A robot drives directly towards a person with a given speed, and then stops right in front of the person. It is then observed how the person reacts. The output from the experiments, was a least squares fit to the minimum stopping distance versus the robot speed up to $5m/s$, for the persons to feel safe. The minimum stopping distance can be calculated as:

$$d_{stop,min} = 0.65 v_{robot} + 0.87m \quad . \tag{2.2}$$

A more general framework for motion in a human environment is introduced in [Kirby et al., 2009]. The framework COMPANION (Constraint-Optimizing Method for Person-Acceptable NavigatION) is a motion framework where social motion conventions are represented as constraints on the navigation. Social conventions are for example that you move to the right side of a hallway, when passing a person. However, conventions are always subject to a trade-off, if e.g. you need to go to a door on the left side of the hallway. Therefore the algorithm optimises the trade-offs of multiple constraints. The paper identifies five particularly important constraints: minimising distance; avoiding obstacles; avoiding persons; and keeping default velocity; and keeping inertia (avoid turning). Each of them have soft and hard constraints. For example the person avoidance consist of a hard constraint that the robot must never attempt to drive through a person. The soft constraint relates to keeping a personal space around the person. The personal space is represented as a potential field a associating a cost to being too close to a person. The potential field is a summation of two Gaussian distributions. The A* algorithm is used to optimise the constraint trade-offs. Results indicate that the robot is able to behave in a predominantly social manner in simulation. A similar approach that consider the state of people in the environment is presented in [Lam et al., 2010, Lam et al., 2011], where rules that guarantee safe and smooth navigation. They define the following rules:

1. Collision Free Rule: avoid collisions
2. Interference Free Rule: avoid people's personal spaces
3. Waiting Rule: if entering a personal space, then wait
4. Priority Rule: if more robots are present, they are given priority
5. Intrusion Rule: robots should not intrude other robot's working spaces
6. Human Rule: humans have the highest priority

For planning a path, the Nearness Diagram (ND) (see Section 2.2.2) together with potential fields, are used. The algorithm is successfully tested on three robots with different priorities. Another optimisation framework has been adopted in [Foka and Trahanias, 2007], where POMDP's are used to optimise the reward associated with avoiding obstacles and reaching a goal position. The environment is split into a static and a dynamic reward grid map corresponding to static and dynamic obstacles and goals. The actions that the POMDP is able to chose between is all possible rotations together with a slow down or increase of speed. The system is demonstrated to work in an entrance hall, where it avoids collisions with people on collision course.

Similar work is done in [Sasaki and Hashimoto, 2006], where a robot moves along frequently used paths in the environment. This is done to act similarly to the people in the environment, and thus provide minimal disturbance to the people. The avoidance of collisions is not explicitly taken into account, but in areas with many people, where avoiding disturbance of people would limit the activity of the robot, this method is useful. The robot, however, does not operate as one autonomous entity, but is provided with much information from the intelligent space it operates in. Sensors embedded in the environment provides information about the robot's position, peoples position and walking paths.

**Approaching and passing in a sociable way:** The opposite to a robot following people or moving along the same tracks as people, is a robot that approach, move towards, or pass people. In [Althaus et al., 2004] it is investigated how a robot should approach a group engaged in interaction. The authors designed a robot along with a software algorithm designed to mimic human-human approaching and interaction in a group conversation. The robot, Robovie, was equipped with sensors including 24 sonars, pressure sensors on the torso, infrared motion detectors and an omnidirectional camera. To engage in an interaction, the robot takes the following steps:

1. Entering the room
2. Approaching the group of people, if there are people present
3. Maintaining the formation with the group of people
4. Moving back to the door, when the discussion finishes

The robot does not verbally interact with the humans, and thus the interaction is limited to passive listening. However, the robot's head turns towards the speaker in the group. For navigation, the robot uses a simple topological map consisting of points of interest to navigate the environment. Points include doorways, corridor crossings, and charging stations. The motion is based on a proportional controller strategy, where the distance to the persons are kept constant and a heading towards the middle of the group. The commanded velocity $v$ is calculated as:

$$v_j = k_{dist}(d_j - D_{inter}) \tag{2.3}$$

$$v = \frac{1}{N}\sum_j v_j \exp^{-c(d_j - D_{inter})} \tag{2.4}$$

where $v_j$ is the desired velocity with respect to one person, $d_j$ is the distance of person $j$, $D_{inter}$ is a desired interaction distance and $k_{dist}$ is a constant. $N$ is the number of persons and $c > 0$ is a decay constant that has the purpose of repairing a formation. In an experiment, the robot was able to move to the persons, reposition itself, when the group formation changed and move back to the door when the conversation finished. The human users reported the robot's behaviour as natural human-like.

In relation to a robot that passes a person who is approaching, [Pacchierotti et al., 2006b, Pacchierotti et al., 2005] describes a study where persons are to pass a robot driving forwards with a speed of $0.6 m/s$ in a hallway. The test subject walks along the hallway and the robot passes at different distances. Afterwards the test subjects are asked how comfortable they felt. This results in an overview over the best behaviours, where it e.g. can be seen that the robot should signal well in advance to which side it wants to pass the person. A figure of the scenario is seen in Figure 2.7.

The above described algorithms are simplified to only cover only one specific task, since they consider either approaching, passing or following single persons. In a real

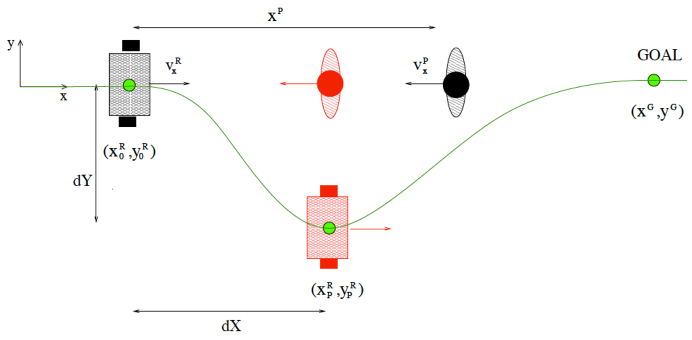


Figure 2.7: The robot passing a person in a hallway scenario. At the distance $X^P$ from the person, the robot deviates to the right to a maximal lateral of $dY$. After passing the person, the robot moves back to the original path. The figure is taken from [Pacchierotti et al., 2006b].

world there could be many persons coming from all directions, and also other obstacles, that needs to be taken into account. This is, for example, done in [Sisbot et al., 2006, Sisbot et al., 2008, Sisbot, 2008], where a completely different approach is taken. The environment is transformed into a potential field grid, where the robot has to move in the low areas. For example around a person, there is a top, which makes the robot go around the person. The combined grid is a combination of several different grids; a safety, a visibility, and an arm comfort grid. An example of such a grid is seen in Figure 2.8. An optimal path for the object is found using the A* algorithm. This approach, however, assumes that all positions of persons are know in advance, and that they do not move, while the robot executes the trajectory.

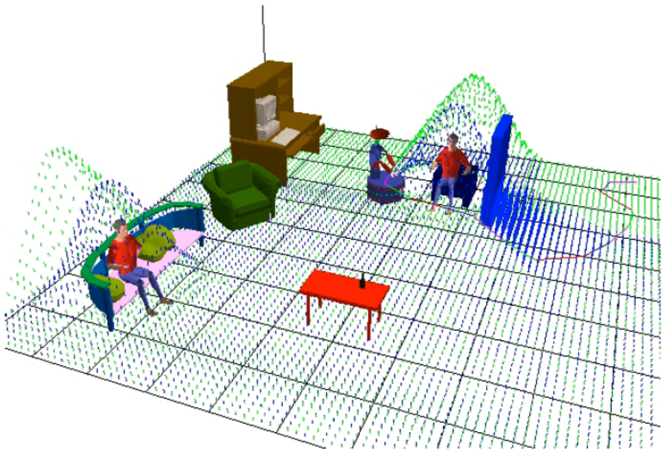


Figure 2.8: The safety and visibility grid introduced in [Sisbot, 2008].

**Robots in the wild:** A discussion of robots that move in human environments is not complete without mentioning some robots, that have actually been set at large in human environments. Although the focus of the works is not so much motion, the still presents some interesting studies of robots that have been moving autonomously in human environments. Rhino, described in [Burgard et al., 1998] was a museum tour guide robot, that guided visitors at "Deutches Museum Bonn". A similar robot was Minerva that was also a museum tour guide robot described in [Thrun et al., 1999]. Both Rhino and Minerva was able to localise itself, plan trajectories in the environment, avoid obstacles and interact with visitors. In [Michalowski et al., 2007], the robot Grace moves around at the conference AAAI 2005 trying to find a person with a pink hat. More recently ACE ([Bauer et al., 2009]) have navigated successfully and asked the way through the streets of Munich. Although these robots are very capable in mapping, localisation and navigation, they lack the ability of being more sociable, i.e. taking human social conventions into account.

**Summary of human-aware motion**

The simple person following experiments in [Gockley et al., 2007, Yoshimi et al., 2006] was performed no more than a couple of years ago, and thus proves that robots moving around humans is only in its infancy. Furthermore, robots that have been tested in human environments, like e.g. [Prassler et al., 2002, Michalowski et al., 2007, Bauer et al., 2009] seems to move relatively slow. So more advanced algorithms are necessary to make the robot move naturally in a human environment. One common issue with the described approaches for human-aware navigation, is that the motion pattern of the robot are static relative to the person, i.e. the motion pattern is always the same. However, for the scenario presented in this thesis, where a person might, or might not, be interested in interaction, the robot needs to be able to adapt the motion pattern accordingly. Therefore it has been chosen to develop a new strategy based on the theory from [Hall, 1963] about how humans position themselves relative to each other. This theory is enhanced with results from [Koay et al., 2005, Sverre Syrdal et al., 2006], which show the direction from which, it is most comfortable to be approached. These results, concerning social acceptable motion, are used together with the idea, from [Sisbot et al., 2008, Sisbot et al., 2005], about using potential fields to guide the motion, to develop an algorithm for social acceptable human-aware motion. However, the human comfort grid from [Sisbot et al., 2006] attempts to keep the robot in front of people and visible at all times. The paths, that this generates may be very unnatural due to attempting to stay visible to people. Thus the method needs to be further developed to take this into account.

## 2.2 Navigation, Trajectory Generation and Motion Planning

Planning of robot collision free trajectories or paths is a more established field of research than HRI. Consequently there exists numerous algorithms for all types of planning that ranges from global planning to local obstacle avoidance and from high dimensional to low dimensional spaces. But even though it is a relatively simple problem formulation, to plan a path from A to B without colliding with obstacles, it is often computationally hard [Siciliano and Khatib, 2008, p.109]. Thus, there is still development for specific problem areas, such as e.g. increasing planning speed, taking into account moving obstacles, increasing the probability of finding a path faster if one exist, and also within more specific application areas like robots in human environments or motion planning for manipulation tasks. The latter is characterised with high number of degrees of freedom, and thus high dimensional search spaces, which often makes calculations intractable and only suited for off-line planning. Good overviews over planning algorithms can for example be found in the books [LaValle, 2006, Thrun et al., 2005, Choset et al., 2005, Siciliano and Khatib, 2008].

The main focus of this thesis is not to develop planning algorithms, but consider motion for a robot in a human environment. Therefore this section only seeks to give an overview over the most important planning algorithms. However, state-of-the-art RRT based motion planning is explained a bit more in detail, since it is used in the thesis. The following overview is mostly based on the above mentioned books. Many systems uses a global planning algorithms to find out which way to go, together with a local obstacle avoidance technique. Therefore an overview over the mostly used existing obstacle avoidance techniques are also given at the end of the section.

### 2.2.1 Planning Algorithms

A brief overview over different algorithms for planning trajectories, is given here. Originally the planning algorithms had focus on optimising the trajectory, but as computer power has increased, algorithms that can use the power of the computers have been developed for planning more complex trajectories. There are three prevalent methods for this; the grid based algorithms, the graph based algorithms, and the sampling based algorithms. In many cases a prerequisite for planning, is having a map. Thus mapping is an important area. For example Simultaneous Mapping And Localisation (SLAM) (see e.g. [Durrant-Whyte and Bailey, 2006]) have had much research interest. However, mapping is not important for the work described in this thesis, and is thus not described further.

### Optimisation based approaches

Traditionally trajectory planning techniques involved solving large optimisation problems to find the optimal trajectory. This is illustrated well in [Betts, 1998], who presents a method for solving a very general trajectory optimisation problem. The general problem is a multidimensional system, which possibly can change dynamics in different phases, i.e. a hybrid system. There is a set of parameters for each phase, which you can adjust and a number of control inputs. Additionally there are imposed constraints on the system that comes from e.g. obstacles or limitations of the physical systems. The task is then to minimize an objective function, which is a function of the state at each time step in each phase, the parameters at each time step in each phase, and the time itself. These optimisation problems are typically very hard to solve, especially if it is an on-line system and time constraints have to be respected. But the formulation of the problems enables satisfying constraints on the robot motions, like non-holonomic constraints and dynamic constraints, which can be difficult for other approaches to planning. To reduce the complexity of the planning, the planning time horizon can be reduced. This leads to receding horizon control, which is also known as Model Predictive Control (MPC), because a model of the system is used to predict the trajectory, which is optimised. An example is [Bak et al., 2001], a path following robot with velocity constraints, is considered.

### Grid based algorithms

Grid based algorithms overlay a grid on the configuration space. Each grid cell is equivalent to a specific configuration of the robot, and it is either occupied by an obstacle or free. The task is then to find a trajectory that passes only through free grid cells. An important setting here is the resolution of the grid. If a fine grid is used, the search for a trajectory can take long time, since the number of grid cells are large. On the other hand, if having too coarse a grid the algorithm might fail to find passages through narrow spaces. A problem with the grid based algorithms, is that the number of grid cells grows exponentially with the dimension of the configuration space. An algorithm that, for example, can be used to find a path, is the wavefront algorithm [Barraquand et al., 1992]. The wavefront works by expanding a “wavefront” from the goal towards the start configuration. Initially all empty cells are set to a value of 0. Then the value of all cells adjacent to the goal are set to 1. The values adjacent to these are set to 2, and so the wavefront expands. When the start position is reached, the shortest path can be found by following the gradient of the potential field, which has been created by the wavefront.

When there is uncertainty in the mapping, each cell can be associated with a probability of being occupied, and this case the map is denoted an occupancy grid map. Here POMDP’s can for example be used for finding a path that minimises the probability for collision, or optimises other performance criteria [Thrun et al., 2005].

### Graph based algorithms

Graph based algorithms are based on a topological map, instead of the grid based map, which is described above. Planning algorithms work by first defining the topological map (a graph) of the environment, and then plan a trajectory on the graph. The idea is to connect different possible configurations in the environment and then move in a straight line between those configurations. Examples of different types and different algorithms to create graphs include; the visibility graph, Voronoi diagrams, plane sweep algorithms and other cell decomposition techniques [Aurenhammer, 1991, de Berg et al., 2000], [Chazelle, 1987, Edelsbrunner, 1987]. Algorithms for searching for a path in a grid includes Dijkstra's algorithm [Dijkstra, 1959], the A* (A star) algorithm [Fikes and Nilsson, 1971, Pearl, 1984, Hart et al., 1968], and the D* algorithm [Stentz, 1994]. All of these algorithms work by assigning a cost to the motion between two points in the graph and then finding the minimal cost path. The simplest of the algorithms is Dijkstra's algorithm. It works by incrementally building a tree and expanding the vertices of the tree. At each expansion the vertex with the least associated cost is expanded to the vertices, which are connected to the given vertex in the topological map. Thus, when reaching the goal the minimal cost path (i.e. the minimum based on the map, which is not necessarily the minimum cost path in the real world) has been found. The A* algorithm expands on this by adding a heuristic value for the cost to go, and expands the vertex, which have the lowest cost plus cost to go. This can greatly reduce computation time. D* plans a trajectory in a partly unknown environment. It makes assumptions about the unknown parts, like for example that it contains no obstacles. It then finds a shortest path from the current configuration to the goal. When the robot observes new information, it adds this information to the map and, if necessary, plans a new optimal path [Stentz, 1994].

### Sampling based algorithms

The increase in computer power has recently (within the last 15 years or so) given rise to a new generation of algorithms, which have gained widespread use in research and also used real applications. These are the sampling based algorithms, which exploits the increasing computational power to test a lot of possible trajectories or configurations, instead of searching for the one optimal solution. Because of the randomness in the algorithms, sampling based planners do not give any guarantees of finding optimal solutions. Within bounded time, they cannot guarantee to find a solution, even if one exists. But most sampling based algorithms are, however, probabilistic complete, which means that the probability of finding a solution, if one exists, goes to 1 as the number of samples goes towards infinity. Other advantages of sampling based algorithms are that they are especially good for high dimensional search spaces, and relatively easy to implement. A survey over sampling based motion planning and current issues can be found in [Lindemann and LaValle, 2005].

There are mainly two algorithms that are used in research. The first is Probabilistic Roadmaps (PRM), which is also called sampling based roadmaps. PRM's were introduced in [Kavraki et al., 1996] for planning in high dimensional configuration spaces. Instead of constructing a complete topological map, as described above, random configuration samples are used to create the map. It works by sampling a configuration and then trying to connect this configuration to nearby sampled configurations. An illustration of this is seen in Figure 2.9. Here the new sampled configuration, $\alpha(i)$, is connected to the nearby vertices. When the graph is constructed, a path is found by using the same

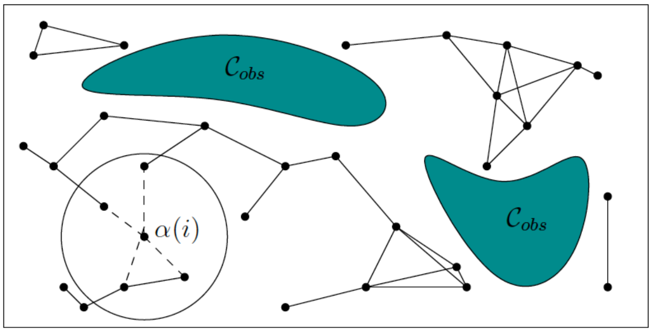


Figure 2.9: A sampling based roadmap, which is constructed incrementally. It is attempted to connect each new sample, $\alpha(i)$, nearby vertices. The figure is from [LaValle, 2006].

algorithms as in the graph based algorithms (see Section 2.2.1).

### Rapidly-exploring Random Trees (RRT)

The other mainly used algorithm for sampling based planning, is the RRT algorithm, which was originally described in [LaValle and Kuffner Jr, 2001]. RRT's belongs to the incremental search type of algorithms, which starts expanding the tree from the start configuration towards a goal. In an RRT, the tree is expanded over the configuration space by sampling a random configuration and extending the nearest vertex towards the new configuration. This is demonstrated in Figure 2.10, where the three vertices to the left belongs to the current tree. A random sample $\alpha(i)$ is then taken. The nearest vertex in the tree is $q_n$, and this vertex is then extended towards $\alpha(i)$. A modified version of the RRT algorithm is used in the work presented in this thesis. Therefore a more in depth description of the RRT is given in Section 3.4.1.

One advantage of the RRT algorithm is that when searching incrementally, it is possible to incorporate kinodynamic constraints into the planning. Another advantage is that

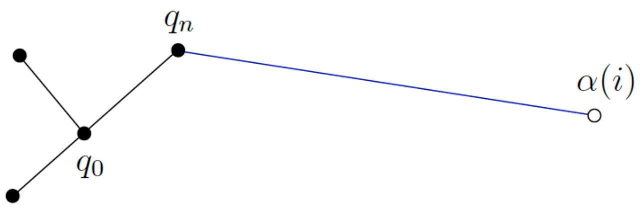


Figure 2.10: To construct the RRT, a new edge, that connects from the sample $\alpha(i)$ to the nearest point in the already constructed tree. The figure is from [LaValle, 2006].

by sampling randomly in the configuration space and extending the nearest vertex, the exploration is biased towards places not yet visited. This can be seen by constructing the Voronoi diagram for the vertices. Larger Voronoi regions are areas which are not well explored and vertices. Since vertex selection is based on the nearest neighbour, it implies that vertices with corresponding large Voronoi regions, are more likely to be expanded. The standard RRT have been modified in many ways for different purposes since it was presented the first time. Some of the modified versions are of relevance to the Ph.D. project, and these are shortly presented here.

It is often desirable to run the planning algorithm in real time, hence requiring bounded solution time. [Ferguson and Stentz, 2006, Ferguson and Stentz, 2007] presents the Anytime RRT algorithm. An anytime algorithm quickly finds a solution, and then keeps improving this solution until there is no more time. First a standard RRT is performed without taking into account the cost. Then a new RRT is performed, where different planning schemes are used to ensure improvement of the planned trajectory.

In human environments, people move around, and are thus not only static, but dynamic obstacles. Methods for incorporating dynamic environments are few, but have been investigated in [Ferguson et al., 2006, Zucker et al., 2007]. The planning with dynamic obstacles is done by "repairing" the RRT when changes occur. This can for example be done by pruning and rebuilding the tree.

RRT's have also been used in environments, where other robots, and sometimes even humans occur. For example, in [Brooks et al., 2009], a dynamic model of the robot and a cost function is used to expand and prune the nodes of the tree. An MPC based approach is taken, where only a small part of the trajectory is executed, while a new trajectory is calculated. When expanding a vertex, a random vertex and a random control input is chosen. The MPC approach for on-line planning is applicable in this thesis, but the random control input sampling does, however, not give a very good result, which is shown in Section 5.5.

The randomness of the sampling in an RRT makes it difficult to guarantee any optimality of the trajectory. Therefore recent research have investigated how to get the notion of optimality combined with the random sampling. Very interesting work is done in [Tedrake, 2009] on combining an LQR algorithm with an RRT. This LQR-tree combines the performance and stability from the LQR algorithm with the RRT algorithm that is good at quickly exploring the configuration space. Similar work has also been done in [Karaman and Frazzoli, 2010], where a variation of the RRT can be used to ensure optimality properties. A disadvantage with these algorithms is though, that they are much slower than a standard RRT. These methods are quite new and have not been studied further in this project.

Investigating the above described algorithms reveals that only [Brooks et al., 2009] is made for human like environments. Furthermore, most algorithms only consider static obstacles and a completely predefined environment in which to plan. Additionally, most RRT algorithms focus on reaching a certain goal, and the algorithm cannot terminate before that happens. This means that they are not well suited for on-line applications. What is needed, in terms of this thesis, is an algorithm, which: is capable of planning in an environment with dynamic obstacles; performs well in on-line applications; and can plan with focus on minimising a trajectory cost instead of reaching a specific goal.

**Nearest Neighbour Search** One topic in RRT based planning is to improve the distance metric, which is about improving the speed of finding the nearest neighbour. The easiest algorithm to implement, is the brute force approach that tests all other vertices to find the nearest. But the brute force approach can be time consuming, and thus other methods are relevant to consider. One such method is the $kd$-tree. The $kd$-tree, which is illustrated in Figure 2.11, works by recursively subdividing the configuration space in each dimension. First the middle sample in the first dimension, is found. This is the top vertex in the tree, and the configuration space is split through this vertex (the vertical line through point 7 in Figure 2.11. Then the two partitions are each split through the middle points in the next dimension (point 5 and 8 in the figure). When all the dimensions have been cycled through, the algorithm continues from the first dimension again. The $kd$-tree has, however, mainly been developed for searching databases. Therefore it does not take into account some inherent properties of planning algorithms, like the dynamic number of vertices and known sampling distribution.

### Summary of planning algorithms

During the last years, sampling based planning algorithms have shown promising results for various planning applications. Especially the RRT algorithm has been applied in many robotics applications. Thus, we have chosen develop this algorithm for use in to planning in human environments. There only exist a few results on sampling based planning algorithms for robots in human environments. To apply the RRT for planning in human environments, it is necessary to account for both the human motion and their

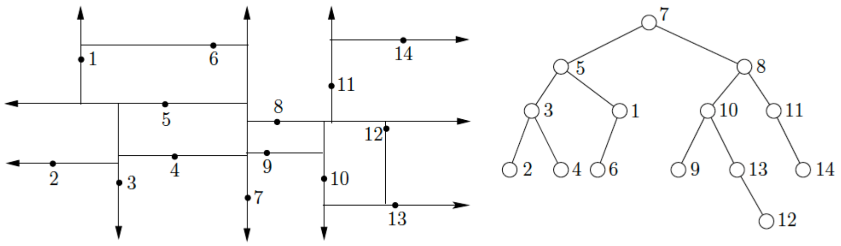


Figure 2.11: For finding the nearest neighbour, a $kd$-tree can be used for more efficient search. On the left is an example of the geometric partitioning of a 2-dimensional space, and on the right is the corresponding graph. the The figure is taken from [LaValle, 2006].

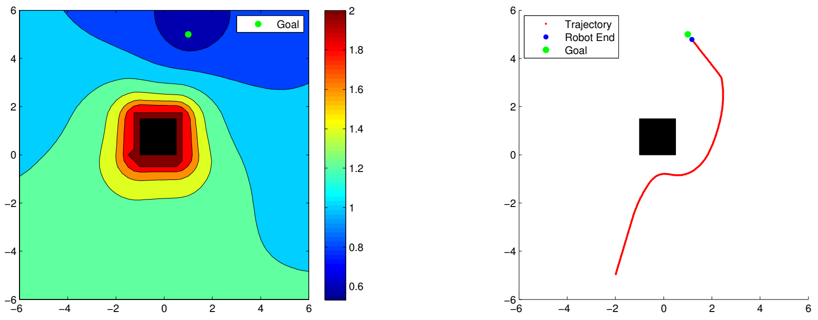
social spaces. Additionally many of the planning algorithms does not take into account the dynamics of the robot, which also need to be accounted for if moving fast in a human environment.

### 2.2.2 Obstacle Avoidance and Dynamic Obstacles

When the surroundings are unknown and unpredictable and obstacles move around, the global trajectory planners often fail to find a collision free path. Actually, the majority of techniques for mapping have been developed for static environments [Siciliano and Khatib, 2008, p.866]. Therefore planning algorithms are often combined with local obstacle avoidance algorithms. There are a number of different widely used obstacle avoidance techniques, such as potential field methods, Vector Field Histogram (VFH), Nearness Diagram (ND), Dynamic Window Approach (DWA) and Velocity Obstacles (VO). The first four assume that the obstacles are static, and the last one are based on the assumption that the obstacles move, and only DWA considers the dynamics of the robot. A short summary of the algorithms is given in the following paragraphs.

**Potential Fields** One of the most used methods are the potential fields. The potential field method uses the analogy that the robot is a charged particle moving in a force field. The goal region will attract the particle, and obstacles repel the particle. The approach, originally developed in [Khatib, 1986] for a mobile manipulator, has been widely applied due to its versatility and simplicity. To illustrate the method, a simple potential field with one obstacle and one goal, has been constructed. This is shown in Figure 2.12, where a robot starts in point $(-2, -5)$, and has to go to the goal at $(2, 5)$. It can be seen in Figure 2.12b that the robot smoothly avoids the obstacle and safely arrives at the goal.

**Vector Field Histogram (VFH)** The vector field histogram was developed by [Borenstein and Koren, 1991]. The VFH approach maintains a polar histogram that represents the confidence of the existence of an obstacle at each direction. The histogram is



(a) The potential field for a simple square black obstacle and a goal.

(b) The robot trajectory from $(-2,-5)$ towards the goal as governed by the potential field.

Figure 2.12: Example of a robot that avoids a simple obstacle (the black square) and arrives safely at the goal (the green dot).

then divided into sectors and the sectors are analysed to find the most favorable direction to move. VFH has been further developed, for example in [Ulrich and Borenstein, 2000] to VFH*, where the A* algorithm is used to verify that the robot is guided around an obstacle.

**Nearness Diagram (ND)** This method is more a methodology to design an obstacle avoidance method, rather than a method itself [Siciliano and Khatib, 2008]. The ND splits the space into regions according to criteria for safety, goal direction, and free walking directions. According to the criteria, different motion strategies are chosen. ND assumes point or circular robots [Minguez and Montano, 2004].

**Dynamic Window Approach (DWA)** The DWA solves the problem by optimizing an objective function, that aims at achieving a goal-directed behaviour while avoiding collisions. The DWA solves the problem in control space and can thus take into account dynamics of the robot. Given a maximal braking acceleration, the subset of control input that can be can canceled before a collision, is found. A control input is then selected from this subset of controls as a trade-off between goal direction, velocity and clearance [Brock and Khatib, 1999, Fox et al., 1997].

**Velocity Obstacles (VO)** VO is an approach, where the velocity of an obstacle is taken into account [Fiorini and Shiller, 1998]. First the velocity vectors that eventually will result in a collision with a obstacle regarded as static, are computed. This will result in a cone as seen in Figure 2.13. This cone is then translated with the velocity vector of the obstacle to obtain the VO. Selecting a velocity outside the velocity obstacles will result in avoiding collisions, if the obstacles keep their direction and velocity. VO work very well

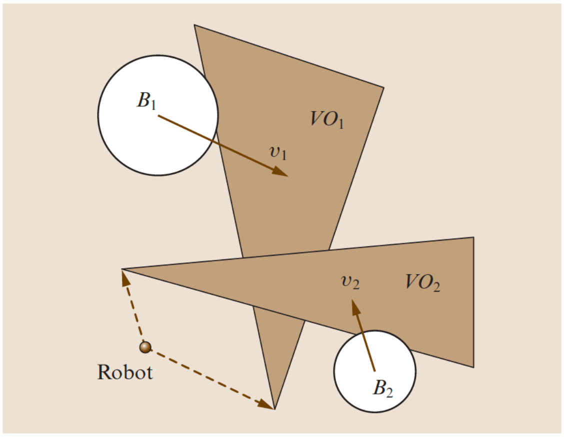
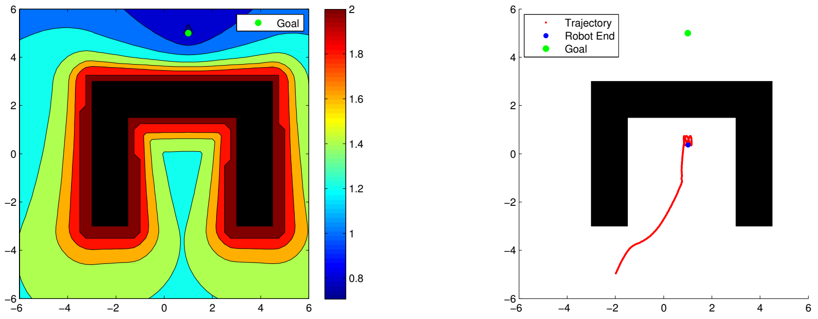


Figure 2.13: The velocity obstacles of two obstacles. If the robot chooses a velocity vector that lies within one of the two cones $VO_1$ or $VO_2$, collision will occur at some point in the future, provided that velocities are kept constant [Siciliano and Khatib, 2008].

in theory, since it guaranties no collision. The problem is that, if the obstacle changes velocity, a collision may occur. In [van den Berg et al., 2008], the reciprocal velocity obstacle is presented. Here it is assumed that other moving obstacles will apply the same algorithm for avoidance and then the velocity is chosen such that the passing happens with no distance in between. This works well in simulations of human environments. However, it would probably not work for real human environments because you cannot assume that other people apply the same avoidance technique as a robot.

**Summary of obstacle avoidance**

One inherent problem with the obstacle avoidance methods are that they all treat the problem locally. This means first of all that they have to be combined with another planning algorithm that tells the obstacle avoidance algorithm where to go. The local way of dealing with the problem also gives rise to other problems. The algorithm might lead the robot to suboptimal paths in a global perspective, because they are not able to re-plan if changes occur in the environment. They also lack the ability to find out what to do if e.g. coming to a too narrow passage, and thus can get stuck. An example of these problems is illustrated in Figure 2.14 for the potential field approach. The obstacle from the above example has been expanded to a U-shape. In this case it is clearly seen in Figure 2.14a that there is a local minimum in the center of the U. In Figure 2.14 it is seen that the result of this is, that the robot gets stuck because the method is not able to re-plan it’s trajectory. Consequently, methods for overcoming this problem have also been studied, but in general, it is not possible to completely overcome the problem.



(a) The potential field.  (b) The robot trapped in a local minimum.

Figure 2.14: Example of a problem that can occur, when using a potential field. The robot can be stuck in a local minimum.

Based on these problems, it has been chosen not to apply these obstacle avoidance algorithms in the project for three reasons. First, the desired output is a combined motion planning algorithm, that solves the motion planning for the robot, and not subsequent levels that give input to each other. Secondly, most of the algorithms does not account for moving obstacles, which humans are. Finally and maybe most important, we do not see people as pure obstacles that just need to be avoided, but as individuals that needs to be treated in a more intelligent manner.

## 2.3 Overall Summary

The goal of this project is to enable a robot to move safe and comfortably in a human environment, while finding out if people are interested in interaction, and move appropriately according to this. To do this, theories from the HRI are combined with navigation and motion planning research. There exist several methods for detecting and tracking persons from simple laser rangefinder measurements, but none of these are also able to find the pose of the persons, which is important for HRI tasks. Researchers have also investigated how robots should learn to behave in human environments, but it has not been investigated how a robot can find out if a person is interested in interaction. Motion around humans have also been investigated thoroughly, both with respect to how a robot should move and implementations of various motion strategies for robots in human environments. Most of the algorithms just consider persons as obstacles, but some algorithms also take into account the social zones of people. However it has not been investigated how to move appropriately, when the person might or might not be interested in interaction.

Navigation and planning algorithms is a well established research field. However, not many algorithms have been applied for mobile robots in human environments. Furthermore, most obstacle avoidance algorithms does not account for moving obstacles, which is the case in human environments. So very shortly, methods that use state-of-the-art planning algorithms combined with knowledge of the human state and social spaces in a human environment, have not been developed.

# 3 Methodology

This thesis uses and extends a range of methods within HRI and robot navigation. In the appended papers, there have been limited space for describing some of the basic algorithms and equipment used throughout the project. This chapter serves to give a bit more detail to some of this.

But first, the SantaBot experiment, which was shortly presented in the motivation (Section 1.1 on page 1) as a start-up experiment for the Ph.D. project, is described more in detail. This is done for better understanding of the experiment and the outcomes, which served as a catalyst for defining the research questions, which serves as the foundation for this Ph.D. project. After this Hall's proxemic theory, which describes how humans position themselves relatively to each other, are described. The Hall zones form the basis of the design of the human-aware motion. After this is a description of the basic CBR method, which in the appended papers, is modified to be able to estimate human intentions. This is followed by a section that elaborates on how typical robot control systems work and how the developed navigation algorithm fits into this. Furthermore the section describes how the very basic RRT algorithm works. Finally, Robotino, our robot that was used during experiments, is described a bit more in detail.

## 3.1 The SantaBot Experiment

The Ph.D. project started out with a social experiment, where a robot was set at large in a public transit space. This was done to see how people reacted to an autonomous robot in their natural everyday environment. The experiment was done as a pilot study to get experience with operating a robot in the real world and to see how people reacted. It is as such not a part of the theoretic contribution to the thesis. But the results are summarized here in the methodology. The results are documented in detail in [Svenstrup et al., 2008], and has later resulted in a video and an extended abstract for the HRI conference in 2009 [Hansen et al., 2009b, Hansen et al., 2009c].

The experiment explored two main elements: 1) demonstration of the ability of a robot to detect, track and follow people in a real world scenario; 2) investigation of peoples perception of the robot, the space and the interaction. So both the technical aspects of placing mobile robots in open-ended environments are investigated, and so is the understanding of the nature and potential of robotic agents in public urban transit spaces.

### 3.1.1 Experimental Setup

The experiment was performed in a combined public shopping area and an urban transit space, giving a very dynamic and diverse experimental environment. In the setup no information about the experiment was given to the persons involved in advance.

The robotic platform used in the experiment was the same, as the one used throughout the Ph.D. project. It is a modified FESTO Robotino® platform, which is further described in Section 3.5 on page 51. To facilitate interaction the robot was dressed as Santa Claus (the experiment was carried out just before Christmas) and equipped with a loudspeaker playing the well known Christmas jingle “Jingle bells” when a person was detected and the robot tried to initiate interaction. The robot can be seen in Figure 3.1, and because of the outfit, we named it “SantaBot”.

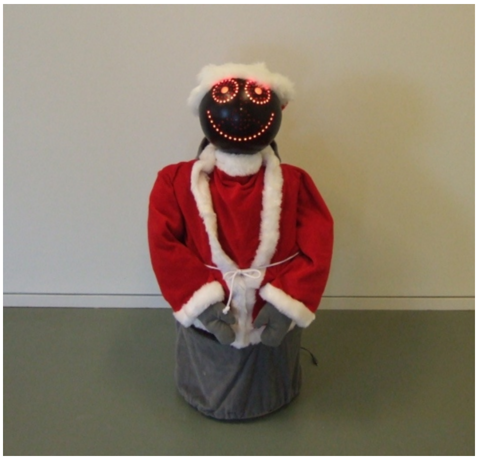


Figure 3.1: The SantaBot outfit.

The robot was programmed to simply follow random people around, to see how they reacted. The outcome of the experiment was evaluated by, questionnaires, video recordings, and in situ observations.

### 3.1.2 Results and Observations from the Experiment

A typical scenario from the experiment is seen in Figure 3.2, where a mother and a child interact with the robot. On the technical side, the experiment showed that the robot was able to detect and track persons in a real world scenario.

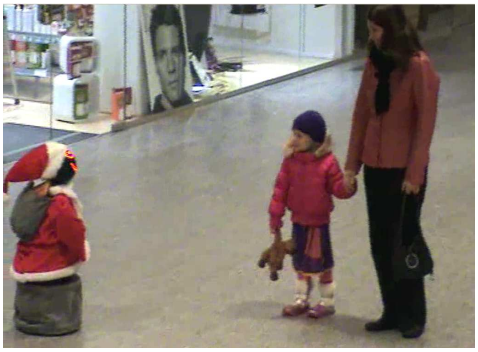


Figure 3.2: A mother and a child interacting with the robot, a typical situation from the experiment.

There were 48 persons, who answered the questionnaire. A summary of the answers to a selection of the questionnaire are displayed in the three histograms in Figure 3.3-3.4.

Regarding questions about how people experienced the robot, the majority (92%) answered that they were positive towards (Neutral to Good) when they first noticed it and it started to approach them. Further, in general people (67% Neutral to Good) did not feel uncomfortable when the robot followed them, though a few persons answered that did not like the robot following them (see Figure 3.3). Of the people questioned 90% found that a robot would fit Neural to Good into a transit space like the “Kennedy Arkaden”, see Figure 3.4.

When asked if they expect robots and what role they would have in transit space in 20 years, 23% expect robots to have a role in entertainment, 40% see robots as assistants, helpers or guides when used in transit spaces and 6% expect them to have a role in surveillance. 50% expect robots to be everywhere in 20 years, 29% only expect a few and 8% expect a situation as today. Since the majority of the respondents were positive towards having a robot entering the transit space. This supports the hypothesis that it is possible to add value to urban transit spaces by putting interacting robots into them. Furthermore, most people feels comfortable towards having the robot in the environment, even when the robot were following them.

Another interesting observation was that people started to interact more when the interaction distance was set to be closer, probably due to the robot getting into the personal zone, where the robot can not just be ignored.

The robot system described here, is a relatively “dumb” first generation mobile agent in a transit space. But in general, the results show that people are overall very positive towards the idea of having robots assisting or entertaining persons in public spaces. So this definitely opens op for further research in how to enable robots to become our “digital

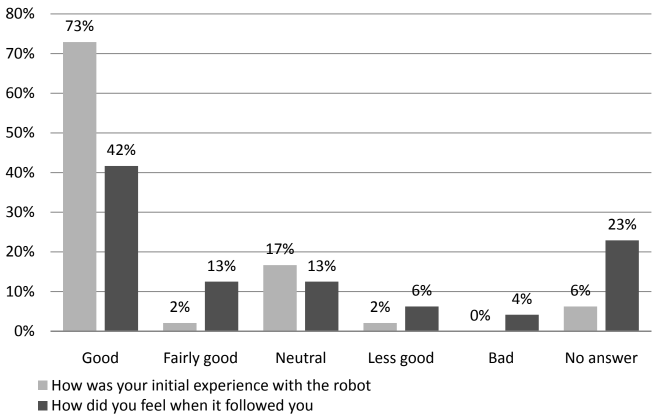


Figure 3.3: Answers from questions on peoples experience with the robot, initially and after the robot started following them.

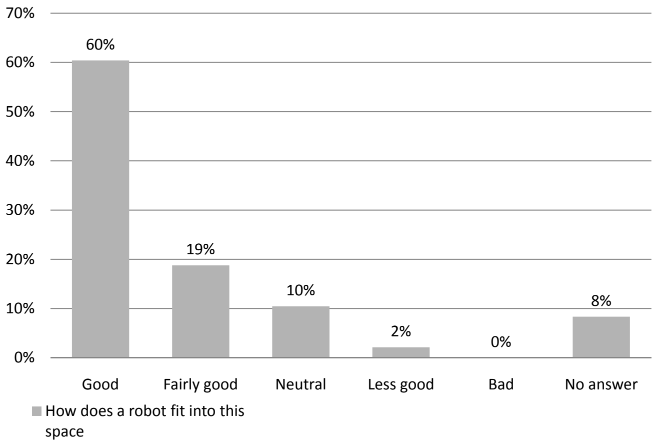


Figure 3.4: Answers from question on peoples opinion regarding the use of robots in transit spaces.

assistant of the future".

## 3.2 Hall Zones

This section briefly describes the Hall zones, which are used through the thesis to govern how the robot moves relative to humans. In [Hall, 1963, Hall, 1966] the anthropologist Edward T. Hall introduced the term proxemics. Proxemics refers to the study of how humans position themselves relatively to others, with respect to distance. Hall studied how a lot of people in public places positioned themselves relative to others, and came up with a model, which is relatively simple. He divides the space around a person into to four zones with an associated distance the person:

1. The intimate zone ($d < 0.45m$), where only interaction like touching or whispering occurs.
2. The personal zone ($0.45m \leq d < 1.2m$), where you have close conversations with people.
3. The social zone ($1.2m \leq d < 3.6m$), is reserved for social interaction in e.g. a larger group.
4. The public zone ($d \geq 3.6m$), is the zone, where there is not really interaction. $3.6m$ is the distance where you start to notice people, and the distance you at least try to keep when walking around.

These zones are also illustrated in Figure 3.5. The defined distances are determined with a basis in Americans, but can change according to the culture of a group of people. For example the distances are typically smaller for Asians, which sometimes can result in peculiar observations, when e.g. an Asian person speaks to a person from a Western country. Asians will step forward to get closer, while the Westerner will step back, resulting in a dance like performance. It has been chosen to use these zones to govern the motion of the robot for mainly two reasons. The zones rules are relative simple, and thus relatively easy to transform to a mathematical formula, which can be implemented on a robot. Secondly, even though the zones were designed with human-human interaction in mind, studies like e.g. [Walters et al., 2005a] and [Koay et al., 2005] have shown to support the use of proxemic distances within HRI as well.

## 3.3 Case-Based Reasoning (CBR)

To estimate a persons interest in interaction, a CBR system is used. This section will describe how CBR works in general, and relate to how it is used in this thesis. The theory here is based on [Kolodner, 1993, Aamodt and Plaza, 1994, Aha, 1998]. CBR is conceptually easy to understand and works much the same way as a human mind. A case

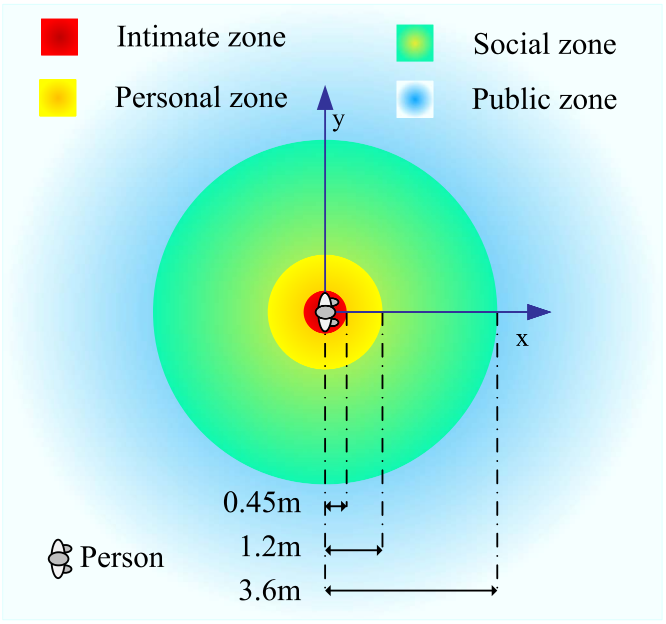


Figure 3.5: The different Hall zones illustrated.

is a set of relevant features extracted from an experience. Like human reasoning, CBR is based on cases rather than based on rules like other machine learning algorithms. Cases from experiences are then continuously stored in a case database like humans remember their experiences. When a new problem is encountered, the task is essentially to recall what happened in previous cases and interpret how this information can be used for the current problem.

Generally there are four steps in CBR: Retrieve, Reuse, Revise, Retain, which is illustrated in Figure 3.6.

**Retrieve:** Is the process of using features from the current problem to find and retrieve similar cases from the stored cases.

**Reuse:** If a good match for the current problem is found in the stored cases, the solution might be directly reused on the current problem. But if a not so good match is found, it can be modified to suit the current problem and then applied.

**Revise:** When the solution has been executed, it is time to reflect on the outcome and see if the strategy should be modified.

**Retain:** When the strategy for the current problem has been revised, the stored cases are updated with the new strategy for the current problem. This part is what constitutes

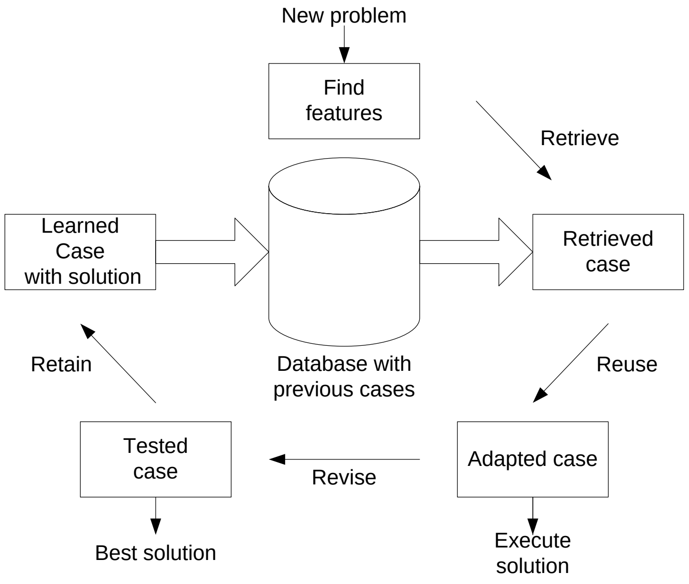


Figure 3.6: The four steps of a CBR cycle: retrieve, reuse, revise and retain.

the learning capability of a CBR system, which makes is improve performance as more experience is gathered.

A slightly modified version of this general CBR framework is used to estimate the intentions of persons in this thesis. The modified version is illustrated in Figure 3.7. When encountering a new person, the retrieve and reuse steps are continuously cycled through, until interaction with the person is done. Only after being done with the interaction, i.e. the person has moved away again, the strategy is revised, and the information is retained and updated in the stored cases.

## 3.4 From Goal to Control Input

A typical task of a mobile robot is to get a goal location and transform this information into motor control inputs, that drive the robot towards that goal. Algorithms that makes this transformation, can be divided into three levels. Although in some systems one or more of the levels are disregarded or integrated into each other. The top level planner uses a map and a goal to find an overall desirable and feasible path from an origin to a goal. The intermediate planner takes the overall path and makes local optimised plans for where to go if sensors observe discrepancies from the overall map of the system. This is also the level, where collision avoidance is handled. These two levels are also recognized in e.g. [Prado, 2007] that states that the control system of a mobile robot generally comprises two different modules: a trajectory planner and a trajecotry tracking controller, although some

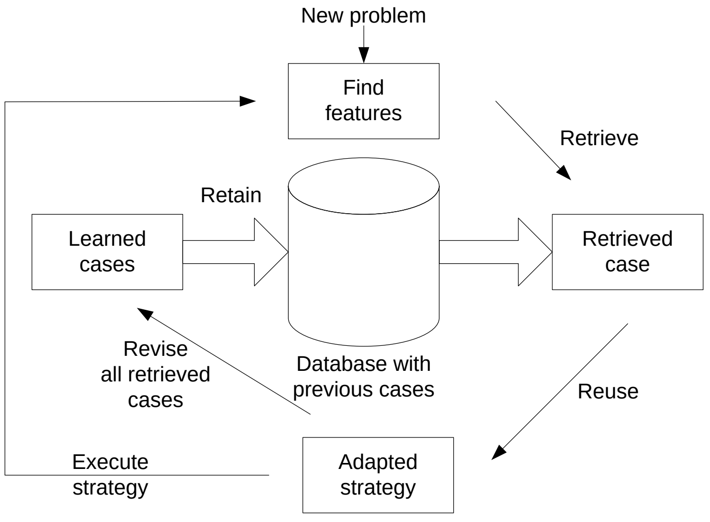


Figure 3.7: The modified version of the CBR system, which is used in this work.

researchers have proposed algorithms that integrate both tasks. But there is an additional lower level, that much robotic research disregards. On the lowest level is the motion control that uses explicit sensor feedback to find the control input that drive the robot towards the desired trajectory. At this level it is typically closed loop controllers, where a given performance can be guaranteed, that calculates the input directly from the sensor readings. At the top level, it is typically more complex, intelligent and time consuming algorithms that tries to generate a good plan. In robotic research, the low level controller is often disregarded and just assumed to be a one to one mapping. This means that it is assumed that you can just send a reference velocity to the robot, and assume that it will be followed. However, this is not entirely correct in most cases. In a real physical world, motion of a robot is always subject to dynamic constraints, which means that it cannot change direction or velocity instantly.

Robotic planning research often only considers the two top levels, which is a global path planner and a local reactive controller of obstacle avoidance. It is assumed that the low level controller is already an integrated part of the robotic system.

The trajectory planning in this thesis comes in the middle or as a replacement of the two bottom levels. It is assumed that some top level planner provides a direction, in which to move. The trajectory planner then both considers how to move locally and uses a dynamic robot model to find exact control inputs. One could argue that the low level controller is actually not a controller, since there is no closed loop. However an increasing number of applications use open loop planners that continuously replans the input in real-time instead of closing a loop by a stabilized closed loop control system[Tedrake, 2009].

### 3.4.1 Sampling Based Motion Planning: RRT

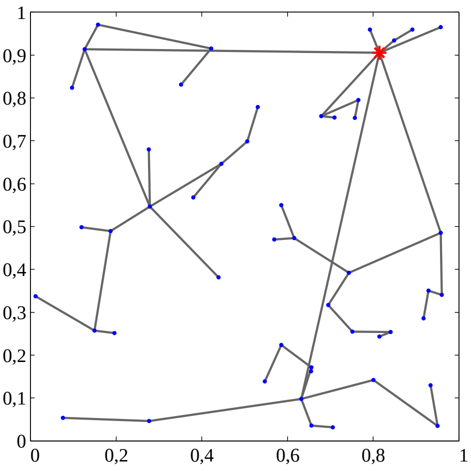
Due to increasing computational power, sampling based motion planning algorithms have gained more attention during the last decade. One very successful algorithm, is called Rapidly-exploring Random Trees (RRT). RRT's are developed specifically for kinodynamic motion planning. Since the work in this thesis uses a modified version of an RRT, this section will shortly describe the basic algorithm.

The basic approach in an RRT is to maintain a tree, where the nodes correspond to connected configurations (vertices) of the robot trajectory. First a random point in the configuration space is sampled. The closest vertex in the tree is expanded directly towards the sampled point. If an obstacle is encountered, the vertex is pruned. This is done over and over again until a new vertex is sufficiently close to the goal, in which case the algorithm is terminated. The algorithm is presented in Algorithm 1. In Figure 3.8a

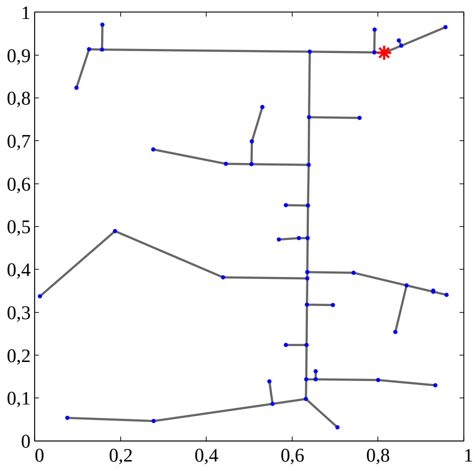
**Algorithm 1** Standard RRT (see [Ferguson and Stentz, 2006])

```
RRTmain()
1: Tree = q.start
2: q.new = q.start
3: while Distance(q.new , q.goal) < ErrTolerance do
4:     q.target = SampleRandomTarget()
5:     q.nearest = NearestVertex(Tree , q.target)
6:     q.new = ExtendTowards(q.nearest,q.target)
7:     Tree.add(q.new)
8: end while
9: return Trajectory(Tree,q.new)
```

a simple RRT with 50 vertices is shown. The start vertex is in the top right corner. A variation of the RRT, which is often used, is where extension towards a new vertex can also be done from an edge. In this case a vertex is placed on the edge from the extension point. This makes the tree only expand orthogonal to existing edges. This approach does however not work very well for kinodynamic motion planning, since it requires $90°$ turns frequently. The RRT is very good at quickly exploring all the configuration space. This is seen in Figure 3.9, where 1000 vertices are added. One thing to note is that there are not many large areas that are not explored well. This is because the extension is biased by the largest Voronoi regions as described in Section 2.2.1.



(a) A simple RRT with 50 vertices.



(b) An RRT where the nearest function extends from edges as well.

Figure 3.8: Two different versions of the RRT algorithm.

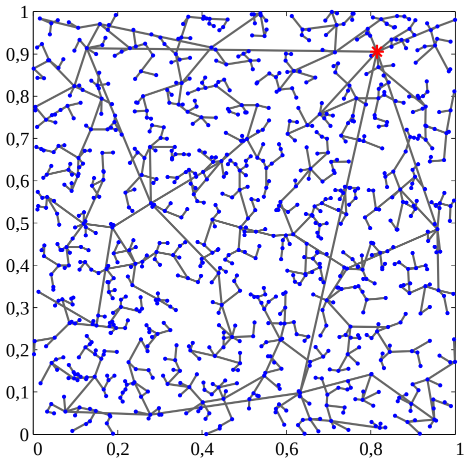


Figure 3.9: A $1000$ vertices RRT, where it can be seen that the configuration space already is covered well.

## 3.5 The Robotino Robot

This section shortly describes the robotic system which has been used during the work on this thesis. The robot and software framework was developed through a Master's thesis project [Kracht and Nielsen, 2007].

The base robot is a Festo Robotino® mobile robot system, which is shown in Figure 3.10. This base robot has been equipped with a head with diodes, which makes it able

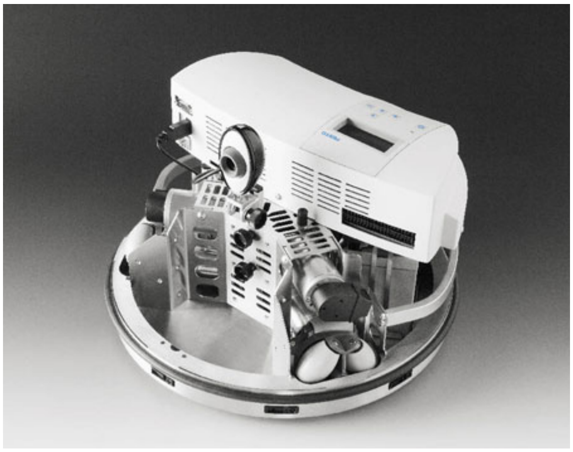
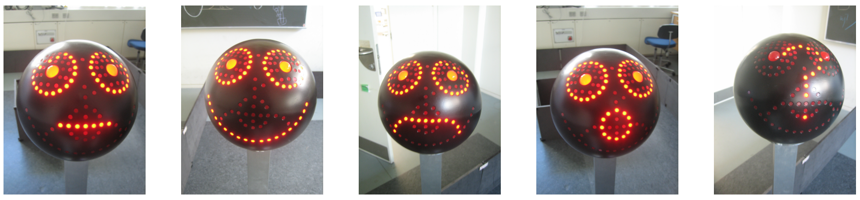


Figure 3.10: The original unmodified version of the FESTO Robotino® robot.

to show different facial expressions for simple communication with the environment. The different facial expressions are shown in Figure 3.11. Furthermore the robot has been



(a) Neutral (b) Happy (c) Sad (d) Surprised (e) Question mark

Figure 3.11: Images of different facial expressions the robot can show.

equipped with an URG-04LX line scan laser range finder, which is used for detection of persons. It has turned out that especially children are very fond of the face, and find it funny to play with the robot. Thus, the framework used in the SantaBot experiment (see Section 3.1 on page 41), where the robot follows a random person, has been used in many occasions to promote university educations at various locations. An example of this is shown in Figure 3.12, where some children plays with the robot.

Figure 3.12: The robot at large in a shopping centre. Children find it funny to play with it.

To control the robot, the robotic software control framework Player/Stage has been installed on the robot [Collett et al., 2005]. All the algorithms presented in the thesis, with exception of the CBR system, have been implemented as plugins for Player/Stage.

# 4 Summary of Contributions

The main contributions of this thesis are in the form of five papers, which are appended after the conclusion of this introductory part. The main contributions from the papers are shortly listed here. The five research questions, which were introduced in the motivation was:

- How can robots detect position and orientation of people by using simple sensors?
- How can robots find out if people are interested in interaction, based on previous experiences?
- How can a general motion control framework be designed to facilitate sociable and comfortable robot motion around a person?
- How can a navigation algorithm be designed to enable robots to navigate through a populated human environment, such that it is safe and natural for the people in the environment?
- How are these abilities evaluated and tested on a prototype robot?

The contributions are described in three sections, which clearly shows the progression of the work presented in the thesis. In the first section, the solution framework to first three research questions, which is published in *Paper A* and *Paper B*, is described. The work is also further documented in[Andersen et al., 2008, Hansen et al., 2009a], which are not a part of the thesis. The next section relates to *Paper C*, where the general applicability of the framework is demonstrated by modifying it into a robotic game. Finally the results from human-aware motion framework is extended to environments with many people. This is what is presented in *Paper D* and *Paper E*.

## 4.1 Adaptive Human-Aware Navigation and Interest Estimation

The starting vision is a service robot driving around in some human environment. When the service robot encounters a person, it must move around that person in an appropriate way. To do this it must have three different capabilities:

- The robot must be able to detect the person, and find out how the person moves.
- The robot must be able to find out if the person wants help (i.e. has interest in interacting with the robot), or the person should be left alone.
- The robot must move naturally, safe and comfortable according to the perceived interest of the person.

### 4.1.1 Pose Estimation

To be able to move around a person in a way perceived as comfortable, not only the position of the person has to be known, but also the orientation and velocity. To do this an algorithm that detect persons from laser range scanner readings is extended to provide velocity and orientation as well. The person detection algorithm was originally developed in [Xavier et al., 2005].

The positions found using this algorithm are fed through a person state estimation algorithm to obtain velocity estimates. All pose and velocity estimation is done in the robots coordinate frame (i.e. from the robot's point of view) and not in a global frame. This is because when moving around humans, it is the relative motion that matters and not if it is in one or an other side of the room. The person state estimator uses a Kalman filter to find the velocity. The state is comprised of the person position, person velocity and the robot velocity. The measurements are the estimate of the person position and the odometry measurements.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_{pers} \\ \boldsymbol{v}_{pers} \\ \boldsymbol{v}_{rob} \end{bmatrix} \quad , \quad \boldsymbol{y} = \begin{bmatrix} \hat{\boldsymbol{p}}_{pers} \\ \boldsymbol{v}_{odom,rob} \end{bmatrix} \quad . \tag{4.1}$$

$\boldsymbol{p}$ denotes a position vector and $\boldsymbol{v}$ denotes velocities, all given in the robot coordinate frame. The $\hat{\boldsymbol{p}}$ is the estimate of the person position from the person detection algorithm and $\boldsymbol{v}_{odom,rob}$ the robot odometry measurement vector.

Because the measurements are done in the robot's coordinate frame and the robot moves, then it turns out that the rotational velocity of the robot, $\dot{\psi}$, causes the Kalman filter to become nonlinear. This is not a desired property. The nonlinear effect can, however, be overcome, maintaining the linear Kalman filter, by introducing a measurement driven Kalman filter [Jun et al., 1999, van der Merwe et al., 2004]. The idea in a measurement driven Kalman filter is to use sensor readings, in this case rotational odometry data from the robot, to drive the process model as an input. This means that the non-linearity is in the input part, $\Gamma\boldsymbol{u}(k)$, and the rotation can be omitted in the state transition matrix $\Phi$. The advantage is that the input is not a part of the Kalman filter equations, and the filter remains a linear filter. The standard discrete state space model is:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \Phi\boldsymbol{x}(k) + \Gamma\boldsymbol{u}(k) && (4.2)\\ \boldsymbol{y}(k) &= H\boldsymbol{x}(k) \quad . && (4.3) \end{aligned}$$

After some calculations, it can be seen that the input should be replaced as follows:

$$\Gamma(k)\boldsymbol{u}(k) = \begin{bmatrix} p_{y,pers}(k) \\ \text{-}p_{x,pers}(k) \\ v_{y,pers}(k) \\ \text{-}v_{x,pers}(k) \\ 0 \\ 0 \end{bmatrix} T\hat{\dot{\psi}}(k) \quad , \tag{4.4}$$

where $\hat{\dot{\psi}}$ is the measured robot rotation from the odometry data.

After the direction of the velocity vector is found, this can be used to calculate the orientation of the person. This is done in an autoregressive filter:

$$\theta(k+1) = \beta\theta(k) + (1-\beta)\arctan\left(\frac{v_{y,pers}}{v_{x,pers}}\right) \quad . \tag{4.5}$$

### 4.1.2 Estimation of Interest in Interaction

To represent the person's interest in interaction, the variable *PI* is used (see definition in Appendix I on page 210). A CBR system is developed to enable the robot to learn to estimate the interest in interaction, which is expressed by the *PI* variable. The operation of the designed CBR system is illustrated in in Figure 4.1. When a person is encountered the specific motion patterns of the person are estimated. The estimates are compared to what is stored in the *Case Library*, which is obtained from previous interaction sessions. The outcome, i.e. the associated *PI* value, of the previous similar sessions is used to update the *PI* in the current situation. This means that if the motion of the current person is consistent, according to what previous persons have done, the robot will get more and more certain about the *PI* value. Thus, the robot is able to estimate the person's interest in interaction. During the interaction session, the motion patterns of the current person are stored in a *Temporary Cases* database. When the interaction session with the person is over, the robot has new knowledge from this session, i.e. it is known if the person wanted to interact, or not. This knowledge together with the Temporary Cases are used to revise the *Case Library*. For example, if the interaction session resulted in interaction, the cases in the Case Library, which are similar to the ones just experienced in the Temporary Cases, are assigned to a larger *PI* value. This approach makes the robot able to learn from experience to become better at interpreting the person's interest in interaction.

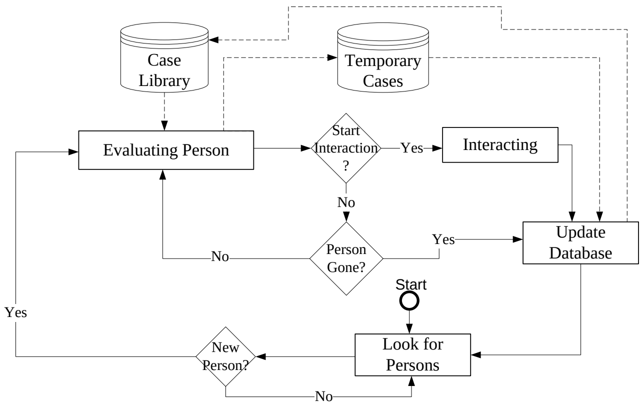


Figure 4.1: First the robot starts to look for persons. When a person is found, it is evaluated if the person wants to interact. After a potential interaction, the robot updates the knowledge library with information about the interaction session, and starts to look for other persons.

### 4.1.3 Human-Aware Navigation

Motion around a human is based on a potential field. A potential field, in which the robot must seek towards the lower parts, is created around the person. The creation of the potential field function is based on proxemics, i.e. how humans position themselves relative to each other, which is described by [Hall, 1963]. Proxemics is further described in Section 3.3 on page 45.

The potential field is made adaptive, such that if it is most likely that the person do not wish to interact (i.e. $PI \approx 0$), the robot should not violate the person's personal space but seek towards the social or public zone. On the other hand, if it is most likely that the person is willing to or interested in close interaction with the robot (i.e. $PI \approx 1$), the robot should try to enter the personal zone in front of the person. The potential field is comprised of four bi-variate Gaussian distributions, which enables the robot to have appropriate behaviour:

**Attractor** distribution, which is a constant negative potential used to attract the robot towards the person.

**Rear** distribution, which ensures that the robot does not approach a person from behind. Therefore it is only and is only evaluated if the robot is behind the person.

**Parallel** distribution, which is an adaptive distribution used to control in which area the robot should move. Since it controls the motion in front of the person, it is only evaluated in front of the person.

**Perpendicular** distribution, which is also adaptive and works in cooperation with the parallel distribution.

The parallel and perpendicular distributions are identical, but rotated $90°$ relative to each other. The shapes of the four Gaussian distributions are illustrated in Figure 4.2. All four

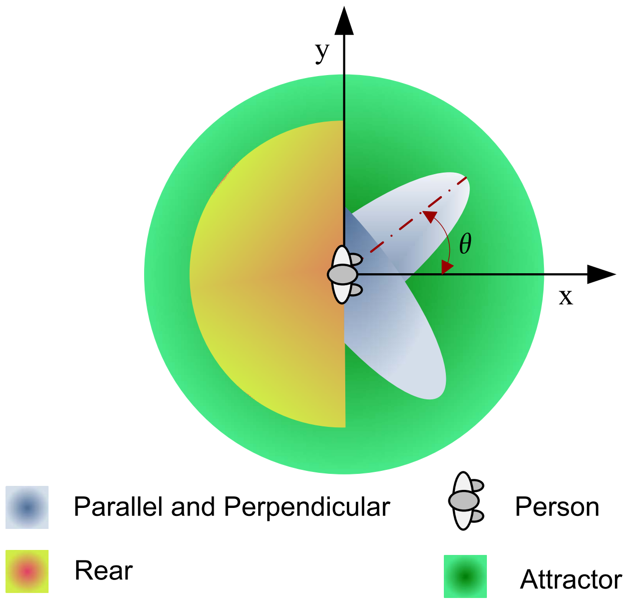


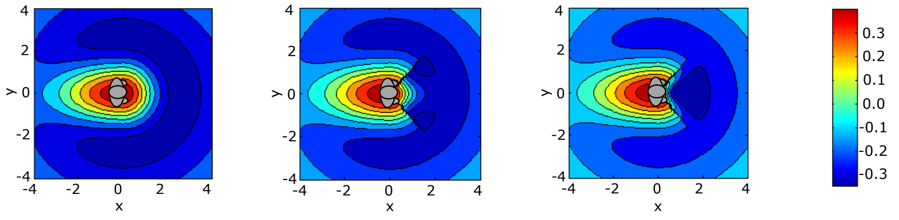
Figure 4.2: The Gaussian distributions, that determine the potential field, in which the robot moves. Note that the rear distribution only is evaluated behind the person, and the parallel and perpendicular are only evaluated in front of the person.

potential functions are implemented as normalised bi-variate Gaussian distributions, and the potential field value in the point $\boldsymbol{x}$ can thus be calculated as:

$$f(\boldsymbol{x}) = \sum_{k=1}^{4} c_k \exp\left(-\frac{1}{2}[\boldsymbol{x}-\mathbf{0}]^T \Sigma_k^{-1} [\boldsymbol{x}-\mathbf{0}]\right) \quad , \tag{4.6}$$

where $k = 1 \ldots 4$ is the different distributions, $c_k$ is a normalising constant and $\boldsymbol{x}$ is the position relative to the person, where the potential is evaluated. $\Sigma_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$ is the 2x2 covariance matrix of the distribution, which determines the shape and rotation. For the parallel and perpendicular distributions $\Sigma_k$ can be calculated such that the desired rotation angle $\theta$ is satisfied.

Although Figure 4.2 illustrates the shape of the potential field distributions, it does not give the full picture about how the combined potential field looks. But these are illustrated



(a) *PI*=0 (b) *PI*=0.5 (c) *PI*=1 (d) Scale

Figure 4.3: Shape of the potential field for (a), a person not interested in interaction, (b) a person considered for interaction, and (c) a person interested in interaction. The scale for the potential field is plotted to the left and the value of the person interested indicator *PI* is denoted under each plot.

in Figure 4.3 for different values of *PI*. Note that the robot always seek towards the lower dark blue areas.

The reference velocity is then calculated as

$$\boldsymbol{v}_{ref} = k\nabla f(\boldsymbol{x}) \quad , \tag{4.7}$$

where $\boldsymbol{v}_{ref}$ is the reference velocity vector, $\nabla f(\boldsymbol{x})$ is the gradient of the potential field and $k$ is an adjustable parameter that controls how aggressive the motion of the robot is.

### 4.1.4 A Robot Game

The framework above is modified in *Paper C* for use in a robotic game. The purpose of the game is to motivate elderly people to do a regular amount of exercise by forcing them to move around. The game rules are simple. A player should receive an object (a ball) from the robot, and try to hand it back. The better the player is, the harder the robot makes it to hand the ball back by moving around. The operation of the robot is:

- If no player is detected, the robots searches for a player by moving randomly around until a person is detected.
- When a player is detected, the robot invites to play a game by approaching the player from the front.
- When the player has accepted to play a game by picking up the ball from the robot, the robot initially moves fast backwards away from the player to get away.
- Then the game is started and the robot keeps avoiding the player with a distance and velocity that corresponds to the estimated skill of the player. When the ball has been handed back, the game is complete.

This behaviour of the robot can be obtained from the framework described above with minor modifications. Instead of *PI*, we introduce *PSI* (Player Skill Indication), which is an estimate of the person's skill. The calculation of the potential field value, as a function of *PI*, is reversed such that $f(\boldsymbol{x}, PSI) = f(\boldsymbol{x}, 1 - PI)$ . This means that, if it is a good player (i.e. high *PSI* value) approaches the robot, the robot moves away like the person was not interested in interaction. And if it is a player with low skill, the robot approaches the person to make it easier to hand over the ball.

## 4.2 Trajectory Planning in Dynamic Human Environments

The algorithms outlined above focus on on-to-one interaction. To facilitate deployment in real human environments, it is natural to extend the results to handle more humans in the environment. *Paper D* [Svenstrup et al., 2010a] describes an algorithm for planning a trajectory through a dense human crowd. Some of the work has also been presented in [Svenstrup, 2010]. The photos 1.6 and 1.5 illustrates this problem, and a schematic of this problem is shown in Figure 4.4. The task is to plan a trajectory from the starting point

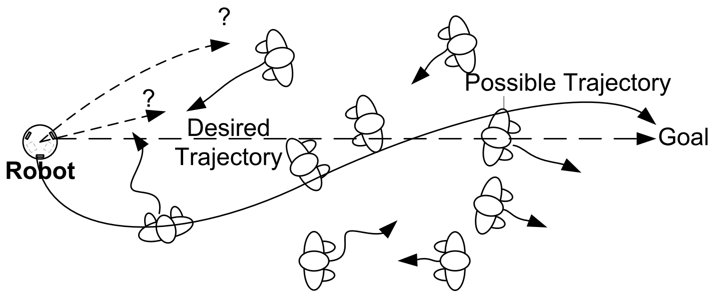


Figure 4.4: Illustration of a scenario, where a needs to navigate through a dense human crowd.

and towards a goal somewhere ahead. The objective is not to exactly reach the goal, but to get towards it. To plan a trajectory through this environment, there are three objectives:

- The robot should stay within a defined area of operation, e.g. stay somewhere near the centre of the street.
- The robot should avoid collision with the humans in the environment
- The robot should get forwards towards the goal.

To do this a potential field can be used to describe where the robot should move. First, a potential field related to the position in the environment is defined. It could be defined any way that suits motion in the specific task environment. In this case we define it as:

$$g_1(\boldsymbol{x}(t)) = c_y y^2(t) \tag{4.8}$$

where $g_1$ is the value of the potential field, $\boldsymbol{x}(t)$ is the position of the robot and $y$ is the deviation from the centre of the street. The potential field described above is used to define where the robot should move relative to humans is also used here. To take into account all the persons, the potential field in Eq. (4.6) is summed over all the persons in the area:

$$g_2(\boldsymbol{x}) = \sum_{j=1}^{p}\sum_{k=1}^{4} c_k \exp(-\frac{1}{2}[\boldsymbol{x}-\boldsymbol{0}]^T\Sigma_k^{-1}[\boldsymbol{x}-\boldsymbol{0}]) \tag{4.9}$$

Since the persons move, this potential field changes over time, and is thus also a function of the persons' positions over time $\mathcal{P}(t)$. Lastly there is a potential $g_3$ that drives the robot forward, and makes sure that the robot does not turn around. It does not really matter how the robot moves along the trajectory if only the ending position is as desired, so $g_3$ is only applied to the end of the trajectory.

Contrary to above, the desire is not to move towards the lowest part of the potential field, but to stay in the low parts and move forwards. So using the three cost functions, an optimisation problem can be formulated:

$$\begin{aligned}
\textit{minimize} \quad & I(\tilde{\boldsymbol{u}}_{0:T}) = \\
& \int_0^T [g_1(\boldsymbol{x}(t)) + g_2(\boldsymbol{x}(t),\mathcal{P}(t))]\,dt + g_3(\boldsymbol{x}(T)) \\
\textit{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\,(\boldsymbol{x}(t),\boldsymbol{u}_t) \\
\textit{where} \quad & g_1(\boldsymbol{x}(t)) = c_y x_2(t)^2 \\
& g_2(\boldsymbol{x}(t),\mathcal{P}(t)) = \\
& \sum_{j=1}^{p}\sum_{k=1}^{4} c_k \exp(-\frac{1}{2}[\boldsymbol{x}_{\boldsymbol{1:2}}-\boldsymbol{\mu}_j]^T\Sigma_{j,k}^{-1}[\boldsymbol{x}_{\boldsymbol{1:2}}-\boldsymbol{\mu}_j]) \\
& g_3(\boldsymbol{x}(T)) = c_{e1}\exp(c_{e2}(x_1(T)-x_1(0))) + c_\theta x_4^4(T),
\end{aligned} \tag{4.10}$$

The combined potential field landscape for an example with five persons and three example trajectories can be seen in Figure 4.5.

Taking a direct optimisation approach to this has proved infeasible. So the approach to solve the problem has been a modified RRT algorithm. The RRT algorithm is modified in the following ways:

- The planner runs in configuration-time ($\mathcal{C}-\mathcal{T}$) space, where moving obstacles are static, i.e. the motion of the persons are taken into account.
- Person motion models are used to predict trajectories of persons

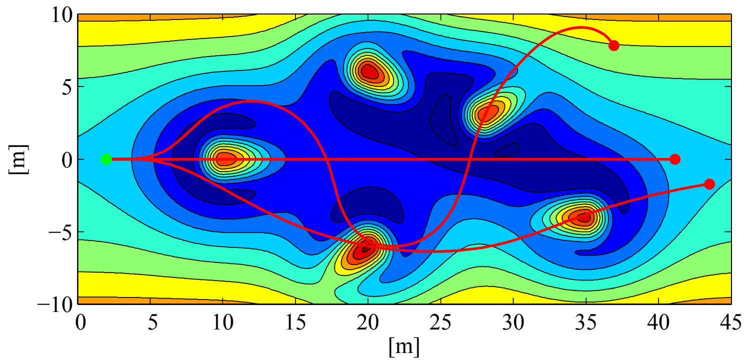


Figure 4.5: Person potential field landscape, which the robot has to move through. The robot starting point is the green dot at the point $(2, 0)$. Three examples of potential robot trajectories are shown.

- When expending the RRT, a controller based approach is taken to extend a vertex towards a newly sampled vertex.
- A second order dynamic motion model of the robot is used to incorporate the robot dynamics when the robot motion is predicted.
- RRT vertices are pruned where the cost is too high, i.e. close to the persons.
- A "best trajectory" is chosen based on the cost of traversing the trajectory, and not based on reaching a goal.
- Using a Model Predictive Control (MPC) scheme, a new "best trajectory" is calculated on-line, while executing the planned trajectory. After a short time period, the trajectory is replaced by the new "best trajectory". And again a new RRT is initialised by seeding with the new "best trajectory".
- The environment potential changes over time according to the desired destination, which also make the algorithm robust to local minima.

An example of a performed RRT on the area in 4.5 can be seen in 4.6, where the green trajectory is the trajectory with the least cost.

A problem with using an RRT is that each time a new vertex have to be added, all other vertices have to be searched through to find the nearest, which is then extended. That means that when the number of vertices become large, the process of finding the nearest neighbour takes a large part of the computation time, and thus makes the algorithm slower. This problem is addressed in *Paper E* ([Svenstrup et al., 2011]), where an algorithm that minimises this computation time of this nearest neighbour is proposed. The algorithm is based on subdividing the configuration space into a number of smaller boxes, and then only a relevant number of boxes needs to be searched. This significantly reduces the computation time. The method is illustrated in Figure 4.7. Where the red dot is a new

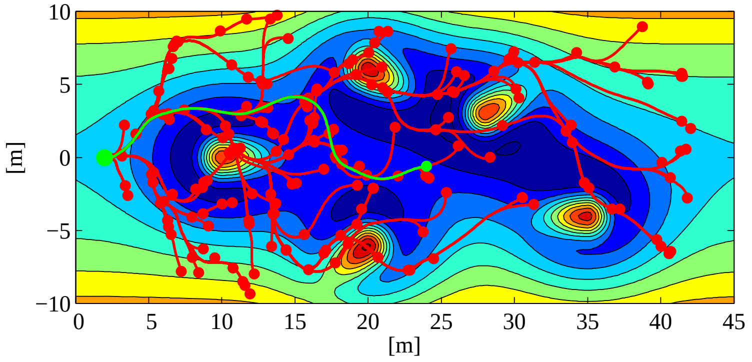


Figure 4.6: An RRT for a robot starting at $(2, 0)$ and the task of moving forward through the human populated environment. Only every 10th vertex is shown to avoid clutter of the graph. The vertices are the red dots, and the lines are the simulated trajectories. The green trajectory is the least cost trajectory.

vertex, which needs to be added. To find the nearest neighbour, first the red shaded box is searched, then the green and so on. If a neighbour that can be guaranteed to be closer than any in the outer boxes, then the algorithm is terminated without having to search through all vertices.

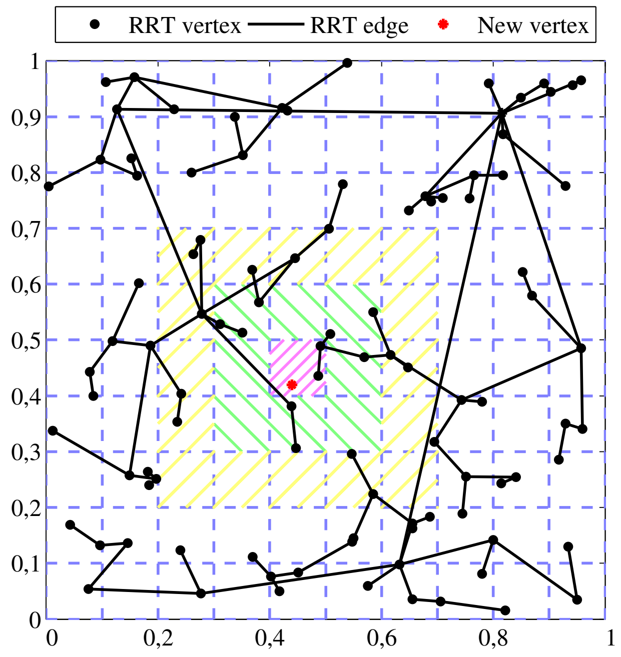


Figure 4.7: The configuration space is split in a number of boxes in each dimension. A grown tree has to be extended with a newly sampled vertex (the red dot). The nearest boxes to the new vertex is shaded in different colours.

# 5 Comparative Evaluations

This chapter evaluates the developed methods and algorithms in comparison with other state-of-the-art research, which is described in Chapter 2. Where it is possible, our methods are compared directly to the performance of other methods. However, most of the work does not have any benchmark performances to compare with, and in those cases, the relation to other work is described. In the following, each of the subsystems; human detection and tracking; intention estimation; human-aware motion; trajectory planning; and the nearest neighbour search are compared to other work.

It is worth noticing that each of the developed subsystems are decoupled, and can thus be combined with other systems as well. This means that if researchers for example find a better way of estimating intentions, it is easy to substitute this part of the system, or if a better way of representing the social spaces as potential functions is found, it is easy to replace the potential field calculation in both the human-aware motion planner and the trajectory planner.

## 5.1 Human Detection and Tracking

We will not evaluate the effectiveness of the tracking algorithm itself, since this has not been the focus of Ph.D. project. There exist many results on different tracking algorithms (see Section 2.1.1), and the purpose for this work was to use one that worked appropriately. Instead we have further developed the algorithm to be able to estimate pose of a person as well. Estimating pose of persons is essential for proper motion around humans. The novelty of the work is to be able to estimate the pose of persons with simple laser range measurements, and not relying on more advanced vision based approaches. For example vision systems to obtain pose estimates require advanced image processing, which is often not applicable to small robots. Several researchers have, in addition to the position of a person also included velocity and orientation [Zender et al., 2007, Althaus et al., 2004, Pacchierotti et al., 2005, Hoeller et al., 2007]. However, all of them uses the direction of the velocity vector as orientation. This is applicable for relative high velocities, but when people are moving slow or standing still, it is a very error prone approach. If, for example a person is moving slightly backwards while standing still and talking to a person or e.g. the robot, such an algorithm would assume that the person is looking the other way. Such wrong estimate might make the robot perform a completely wrong motion. Our approach, however, takes this into account because the orientation estimate is only changed slowly, when the person is moving slow.

## 5.2 Intention Recognition for Interaction

Regarding estimating the intentions of person's interest in interaction with a mobile robot, we believe that it has not been done before. Most work within HRI assumes that the person and robot already are engaged in interaction, or assumes that the person is interested and thus the robot just moves in front of the person. Additionally there exist many results for motion planning in human environments (see Section 2.1.3). However, a service robot in a future human environment must be able to perform both interaction and motion. Currently there is a gap in research for robots that can make that jump from moving around to interacting. There have been research like [Kelley et al., 2008], that tries to estimate human intentions, but not with respect to if they want to engage in interaction. Currently, the research that comes closest to solving this problem is presented in [Althaus et al., 2004]. This research is partly a user study, where a robot joins a group of people engaged in interaction. However, the robot does not try to find out anything about interest, it just drives towards the group of people, if it sees one. That is where our research, in estimating human interest in interaction, fills a gap. We have chosen to use a Case Based Reasoning approach to solve this problem. This method is easy understandable because it fits well with how humans think; we observe events, compare to what we have experienced earlier, and use that information to reason about the current event. The CBR system have proved to work well for the given robotic system.

## 5.3 Human-Aware Motion

Two complimentary methods for motion in human environments have been developed. One, which treats motion relative to one specific person for interaction purposes, and one for navigating through a crowded human environment. Even though they can be used together on a robotic system, they present two different types of planning in literature. Therefore there are two sections in this comparative study. This section covers an analysis of motion in a human environment relative to humans, and not the trajectory planning in itself.

First an overview over the motion related features of other algorithms is presented. Table 5.1 gives an overview over different algorithms which are presented in Section 2.1.3. Additionally our proposed system is added to be able to compare. Table 5.2 is an extension of the first, with the same reference systems and additional properties.The properties of each system are listed without any discussion about if the specific properties are good, bad or necessary to have. This will depend on the specific application. For example having a global map might increase performance of the system. However, if operating in an unknown environment, the algorithm need to function without having a map. The columns in the table corresponds to the following questions:

- What is the **function** of the robot? Is it tracking/following/guiding/interacting/goal seeking/approaching/passing, or just moving around for the purpose of the motion planning research.
- Does the system take **human motion** into account, or are people treated as static obstacles?
- Does the robot **predict** the human **motion** when planning, or only use the current state?
- Does the system take **social spaces** into account, or are people treated as obstacles (dynamic or static)?
- Does the system estimate **pose**, i.e. is the actual pose measured or it just the velocity direction or no orientation at all.
- Does the system consider, that there might be **interaction** with a person, or is it only motion planning around humans?
- Does the system **learn** either human behaviour or how to behave from experiences?
- Does the paper contain a **user study**? If persons are asked how they feel or they are asked to do something until they feel uncomfortable, and this is then used to derive how a robot should move. Then it is regarded as a user study in this context.
- What is the name or type of the **robot**?
- Does the system need a **global map** to work, or is it based on local world knowledge?
- What kind of **motion** planning **algorithm** is used? Is it RRT, A*, an obstacle avoidance algorithm or maybe controller based.
- Does the robot **plan** trajectories around humans, or does it just react to **avoid** possible collisions with humans? It might also expect humans to avoid, and thus not take collisions explicitly into account.
- Does the system utilize **potential fields** to guide the motion?
- Which kind of **sensors** are used for sensing the environment and the people?
- What is the normal operating **speed** of the robot. Many papers does not explicitly state this, maybe because robots are still notoriously slow movers in human environments.

| Robotic System Reference | Function | Human motion | Predict motion | Social spaces | Pose | Interaction | Learning | User study |
|---|---|---|---|---|---|---|---|---|
| [Gockley et al., 2007] | Follow | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| [Prassler et al., 2002] | Follow | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Nakauchi and Simmons, 2000] | Stand in line | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ |
| [Yoshimi et al., 2006] | Follow | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Takemura et al., 2007] | Follow | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Zender et al., 2007] | Follow | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| [Hoeller et al., 2007] | Accompany | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Martinez-Garcia et al., 2005] | Guide | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Müller et al., 2008] | Goal | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Lam et al., 2010, Lam et al., 2011] | Moving around | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| [Kirby et al., 2009] | Moving around | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| [Foka, 2005, Foka and Trahanias, 2010] | Goal | ✓ | ✓ | ✗ | ✗ | ✗ | (✓) | ✗ |
| [Sasaki and Hashimoto, 2006] | Moving around | (✓) | ✗ | ✗ | ✗ | ✗ | (✓) | ✗ |
| [Shi et al., 2008] | None | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| [Tadokoro et al., 1995] | Moving around | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [Althaus et al., 2004] | Engage in Conversation | ✗ | ✗ | (✓) | ✗ | ✓ | ✗ | ✓ |
| [Pacchierotti et al., 2006b, Pacchierotti et al., 2005] | Passing | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ |
| [Sisbot et al., 2006, Sisbot et al., 2008] | Service | ✗ | ✗ | ✓ | (✓) | (✓) | ✗ | ✗ |
| [Koay et al., 2005, Sverre Syrdal et al., 2006] | Approach | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ |
| [Michalowski et al., 2007] | Play tag | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | (✗) |
| [Thrun et al., 1999] | Museum tour guide | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |
| [Burgard et al., 1998] | Museum tour guide | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ |
| [Bauer et al., 2009] | Navigation | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ |
| Our algorithms | Estimate intent/ plan motion | ✓ | (✓)[a] | ✓ | ✓ | ✓ | ✓ | ✗ |

Table 5.1: This table gives an overview of the different properties of the algorithms described in the state of the art in Section 5.3. $(\cdot)$ means partly or not explicitly considered.

[a]The system for estimating human intentions does not explicitly predict the human motion. However, the trajectory planning algorithm does.

| Robotic System Reference | Robot | Global map | Motion algorithm | Plan/Avoid | Potential field | Speed | Sensors |
|---|---|---|---|---|---|---|---|
| [Gockley et al., 2007] | GRACE | ✗ | Controller | ✗ | ✗ | $< 0,7 m/s$ | LRF |
| [Prassler et al., 2002] | Wheelchair | ✗ | Velocity Obstacles | Avoid | ✗ | Slow | LRF |
| [Nakauchi and Simmons, 2000] | Xavier | ✗ | Markov Model | Plan | ✗ | Slow | Stereo Vision |
| [Yoshimi et al., 2006] | ApriAttenda$^{TM}$ | ✗ | Direction Follow | Avoid | ✗ | Slow | Vision |
| [Takemura et al., 2007] | N/A | ✗ | Pot. Field | Avoid | ✓ | N/A | Stereo Vision,LRF |
| [Zender et al., 2007] | PeopleBot | ✓ | Controller / ND | Avoid | ✗ | $< 0.9 m/s$ | LRF |
| [Hoeller et al., 2007] | RWI B21 and Pioneer | ✓ | Related to A* / DWA | (Plan) | ✗ | $< 0.45 m/s$ | LRF |
| [Martinez-Garcia et al., 2005] | Yamabico | ✓ | Controller | (Plan) | ✗ | N/A | Stereo Vision |
| [Müller et al., 2008] | Pioneer 2 | ✓ | A* | Plan | ✗ | N/A | LRF |
| [Lam et al., 2010, Lam et al., 2011] | 3 robots | ✓ | ND/Pot | Avoid | ✓ | $< 0,6 m/s$ | Stereo Vision,LRF |
| [Kirby et al., 2009] | Simulation | ✓ | A* | Plan | ✓ | N/A | LRF |
| [Foka, 2005, Foka and Trahanias, 2007] | B21r (Lefkos) | ✓ | POMDP | Plan | ✗ | $< 0,9 m/s$ | Vision,LRF |
| [Sasaki and Hashimoto, 2006] | N/A | ✓ | Bezier curves | Plan | ✗ | N/A | External |
| [Shi et al., 2008] | Segway RMP | ✗ | RRT | Plan | ✗ | $< 5 m/s$ | Vision |
| [Tadokoro et al., 1995] | N/A | ✓ | Genetic Alg. | Plan | ✗ | N/A | N/A |
| [Althaus et al., 2004] | Robovie | ✗ | Controller | (Plan) | ✗ | N/A | Vision,IR,Sonar etc. |
| [Pacchierotti et al., 2006b, Pacchierotti et al., 2005] | PeopleBot | ✗ | Controller | Plan | ✗ | $< 0,6 m/s$ | LRF,Sonar,Vision |
| [Sisbot et al., 2006, Sisbot et al., 2008] | Rackham Jido | ✓ | A* | Plan | ✓ | $< 1 m/s$ | LRF,Stereo Vision |
| [Koay et al., 2005, Sverre Syrdal et al., 2006] | PeopleBot | ✗ | ✗ | ✗ | ✗ | Slow | N/A |
| [Michalowski et al., 2007] | GRACE | ✗ | N/A | ✗ | ✗ | Low | Vision,LRF |
| [Thrun et al., 1999] | Minerva | ✓ | Costal planner / $\mu$DWA | Plan/Avoid | ✗ | $< 1.6 m/s$ | LRF, Vision, etc. |
| [Burgard et al., 1998] | Rhino | ✓ | Value iteration / $\mu$DWA | Plan/Avoid | ✗ | $< 0.9 m/s$ | LRF, Sonar, etc. |
| [Bauer et al., 2009] | ACE | SLAM | A* | Plan/Avoid | ✗ | $< 1,4 m/s$ | Stereo Vision |
| Our algorithms | Robotino | ✗ | RRT/Pot. Field | Plan | ✓ | $< 2.0 m/s$ | LRF |

Table 5.2: This table gives an overview of the different properties of the algorithms described in the state of the art in Section 5.3. It is continued from Table 5.1. LRF is Laser Range Finder, and N/A is Not Available

When looking at Table 5.1 and 5.2, it is seen that our methods spans the widest parts of the subject areas in the table. However, this might be due to that the table sum up on the areas, that are addressed in this thesis. But this is still quite broad within motion in human environments. We believe that a robot, that is successfully introduced into dynamic human environments sometime in the future, must possess the skills listed in the tables (and possibly more).

Many approaches uses a global navigation strategy, which means that robot and person motion is tracked in a global static map. This means that you have to have some knowledge about the whole world, which we think will not always be the case for future robots moving around in our environment. Our approach does not require any global information, and is able to move around only considering the motion of the humans in the environment, since the person motion is calculated in the robot reference frame. Because the person motion is known relative to the robot, it also makes it easier to estimate what the intentions of the person with respect to the robot, since it can be related to previous experiments. Other research considering the human motion trajectories [Bennewitz et al., 2005, Bruce and Gordon, 2004, Foka and Trahanias, 2010] only considers where the persons may go relative to a static environment, independently of where the robot is located. An optimal approach would of cause involve usage of all this information. The most important features of our human-aware motion planner is the ability to take into account the social zones of a person, and to take into account a person's estimated interest in interaction, which have not been done in other research.

## 5.4 Trajectory Planning in Human Environments

"*Humans cannot be considered just as objects and cannot be simplified to moving obstacles*" [Sisbot, 2008]. Our method for planning a trajectory through a human environment has one fundamental difference from most other research. The robot does not treat humans as merely dynamical obstacles that need to be avoided. The presented algorithm attempts to move the same way humans do, and thus being more natural, which is essential for the acceptance of the robot in our world. Furthermore the planning algorithm respects persons' social zones.

Most works that take the human motion into account attempts not to disturb people as a criterion for the motion. However, this is not entirely possible in a human environment. People also disturb other people when moving around, so why should robots not be allowed to do it as well. Consider for example the entrance to a train station. If the robot has to wait until there is a free path to be able to get through the doors, the robot will never get through because of the constant stream of people. Thus, some form of disturbance must be allowed. This work does not treat the disturbance explicitly, but the robot tries to avoid getting too close to persons, which results in a trade-off between no motion and close passage. In the train station entrance scenario our proposed trajectory generation algorithm might slow down or halt initially, but the change of the potential field over time, would result in the robot moving through the doors within a short time. The dynamic nature of the planner then adapts to the people's reactions to the robot, while moving through the door. The only other work that, to the extend of our knowledge, accepts that robot trajectories might disturb human motion is [Kirby et al., 2009] and [Sasaki and Hashimoto, 2006]. [Kirby et al., 2009] used constraint optimisation to account for this, and in [Sasaki and Hashimoto, 2006] the robot tries to follow human walking paths to minimise the disturbance. However, shortcomings of these methods are the ability to incorporate the dynamics of the human motion, and to operate autonomously in an unknown space, where just the sensory input on the robot is available. Our system both take the human motion into account and navigates only from sensory input on the robot.

For predicting human motion, our implementation use a simple linear motion model, which assumes a person will continue with the current speed and velocity. More advanced motion models, like e.g. [Bennewitz et al., 2005, Bruce and Gordon, 2004], [Foka and Trahanias, 2010] (described in Section 2.1.2), can of course be used in the framework of our system. But humans do not use complex motion models to predict, where other humans will go. The philosophy is, that this should be the same for robots, that it is not necessary for the robot to incorporate complex motion prediction algorithms. Furthermore, complex motion models can be prone to errors if e.g. a person changes goal. To pass each other, we usually allow a very short passing distance. So even if having a relative good prediction of the goal of another person, a small path deviation from the estimated path, will cause the need to update the planned robot trajectory anyway. Thus a simple linear motion model can be just as good as a complex one.

### 5.4.1 Simulation of the Human Motion

When it comes to the simulation environment for testing the algorithm, the human motion model is a bit more advanced to make the simulation more realistic. It is made such that it should resemble person motion relatively well. The simulation of the human motion is based on each person having a goal they move towards. The person motion is simulated according to the formulation given by Eq. 2.1 from [Fajen and Warren, 2003]. Additionally the people in the simulation can randomly change goals and an additional Gaussian noise is added to the direction of the motion and the velocity of the persons.

## 5.5 Trajectory Planning

This section compares the trajectory planning algorithm from a pure planning point of view. Thus, no evaluation of social spaces or human motion as an obstacle, is done. The requirements for the algorithm was to minimise the probability of collision, while still maintaining a forward speed. There are no such guarantees as time optimality, guarantee of best trajectory, 100 percent certainty of collision avoidance etc. This is because our philosophy is to move around like a person would do, and we also sometimes bump into each other because space is narrow or we are not attentive. It is believed that coming up with guarantees imposes constraints on the systems that humans would not have, and thus limits the performance in the end.

To solve the problem, a modified version of the RRT algorithm, has been used. The contributions related the modification of the RRT algorithm are listed here:

- Formulation of the navigation problem as planning a path through a dynamic potential field.
- New RRT trajectory selection and pruning methods, which are based on a potential field.
- Control input selection strategy, which leads to better covering the configuration space than random control sampling.
- MPC based RRT, which only executes a small part of the trajectory while recalculating a new. Unlike the MPC based RRT paper from [Brooks et al., 2009], in which the algorithm executes the whole trajectory.
- Opposite from traditional RRT planning, the focus is on keeping a desired pace, instead of reaching a goal location.
- Including dynamic obstacles in the RRT by planning in Configuration-Time ($\mathcal{C}-\mathcal{T}$) space .

The trajectory planning algorithm is demonstrated to be able to plan a path in a human environment in Section 4.2. However, it is not compared to other comparable algorithms on the same problem. Therefore the same problem is here solved using two other RRT algorithms. One, where a random vertex is selected to expand, and one, where a random control input is chosen after selecting a vertex to expand. The two algorithms are suggested in [LaValle and Kuffner Jr, 2001] and used in e.g. [Brooks et al., 2009, Shkolnik et al., 2009]. To be able to compare the approaches, it is only the tree expansion strategy, which is changed. The selection of the best trajectory and pruning of the tree is still done according to the developed algorithm. In Figure 5.1 an RRT has been run for the same environment as shown in Figure 4.5. But instead of a random sampling in the configuration space and then expanding the nearest vertex, a random vertex is expanded. The algorithm is a little faster than our proposed algorithm, since it does not have to find the nearest other vertex. However, as can be seen, the tree does almost not expand over the configuration space. This is expected because one of the main advantages of the RRT algorithm is, that it exploits that there is a larger probability of sampling into the larger Voronoi regions, and thus leading to good tree expansion in empty (wrt. vertices) regions of the configuration space. When sampling random vertices to expand, this property of the RRT disappears.

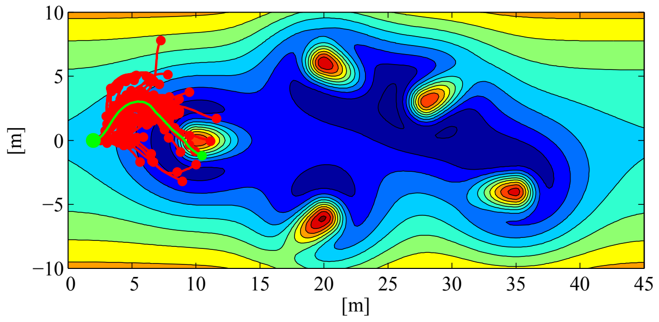


Figure 5.1: The RRT after 2000 expansions, where a random vertex is chosen to be expanded. The tree expands very slow over the configuration space because the random sampling in configuration space is not exploited.

In the next example, the nearest vertex is expanded, but instead of using our control algorithm to guide the robot towards the sampled point, a random control input is selected. The result is shown in Figure 5.2. Comparing to choosing a controller guided control input (which is shown in Figure 4.6), the configuration space is not covered very well, since there are only two major branches of the tree, and the top area above the first person has not been explored at all. The random control input is slightly easier to implement, however it does not force the robot towards the sampled configuration, and thus does not exploit the biasing of the sampling into the largest Voronoi regions. This demonstrates that the developed approach outperforms two other algorithms for these examples. The comparison of our approach to a standard RRT together with the two other approaches is summarised in Table 5.3. The basic RRT, as described in Section 3.4.1, does not take dynamics of obstacles into account and does not account for the non-holonomic constraints of the robot. On the other hand, it covers the configuration space well, and is relatively fast, because it does not need to simulate the trajectory using a robot motion model.

The algorithm has proved to work in simulation, but has sometimes little difficulty in very dense environments. One aspect is, that in the simulations, the persons does not adapt their motions relative to the robot. This makes it more difficult, since it is only the robot that can avoid collisions. But the difficulty in dense populated areas are not unexpected though. Even we, as humans sometimes have difficulties, if we move in too crowded areas. And if so, then we relax on the social conventions and allow each other

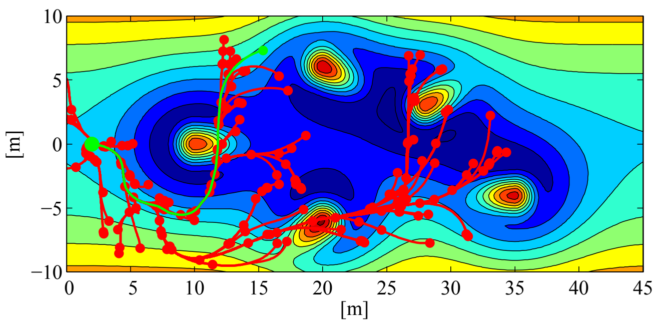


Figure 5.2: An example of choosing random control input. The tree expands better than if choosing random nodes, but it still does not cover the configuration space very well. The lowest cost trajectory is shown in green.

| Method | Dynamic obstacles | Holonomic constraints satisfied | Configuration space coverage | Algorithm speed |
|---|---|---|---|---|
| Basic RRT | No | No | Good | + |
| Random vertex | Yes | Yes | Bad | + |
| Random control | Yes | Yes | Partly | − |
| Our method | Yes | Yes | Good | − |

Table 5.3: Comparison of the basic RRT approach to two modified versions of the RRT, and our version. All of the methods are based on our potential field formulation, but with different sampling and extension strategies. The + and − does not mean explicitly fast and slow, but comparatively slightly faster or slower than the other algorithms in the table.

to move much closer than the personal zone. This behaviour can, for example, be seen at large spectator events, at entrances to shops, in queues, etc.

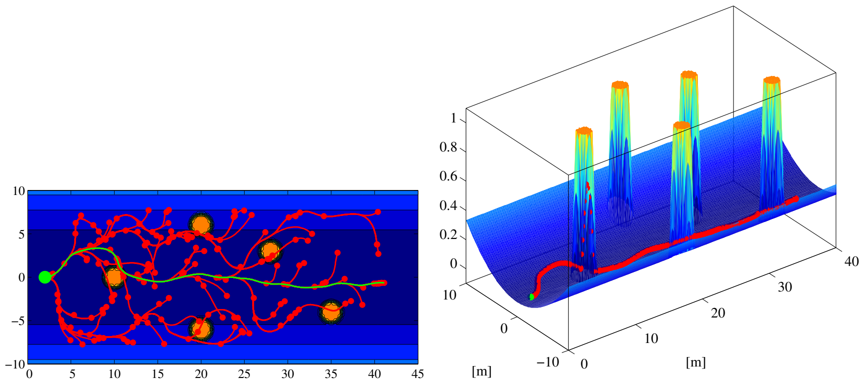
### 5.5.1 An Obstacle Avoidance Perspective

Although the algorithm is developed for motion in a human environment, it can easily be reformulated to a pure obstacle avoidance problem with either static or dynamic obstacles. This enables the algorithm to be used for other robotic applications than just motion in human environments. To convert the algorithm to a pure obstacle avoidance algorithm, the only thing that has to be done is transforming the potential field. The potential field is formed on the basis of human social spaces, which are complex in relation to obstacles. Transforming the potential field can be done by simply assigning a constant value, when closer than a safety distance to an obstacle. If further away, 0 is assigned. However, the shape of the potential field that governs the robot in the environment is maintained. Thus we reformulate $g_2$ from Eq. 4.9:

$$g_2(\boldsymbol{x}) = \begin{cases} c & \text{if} \quad d_{obstacle} \leq d_{safety} \\ 0 & \text{if} \quad d_{obstacle} > d_{safety} \end{cases}, \tag{5.1}$$

where $d_{obstacle}$ is the distance to the obstacle, $d_{safety}$ is the minimum safety distance, and $c$ is a positive constant.

To demonstrate this, the pedestrian street environment is transformed into an obstacle field. For these experimental purposes, the RRT algorithm is run using a safety distance of $1m$ with moving obstacles, and a safety distance of $2m$ with static obstacles. The result of the simulation with moving obstacles and a safety distance of $1m$ can be seen in a 2D and 3D plot on Figure 5.3. Since the obstacles are moving, some of the trajectories seem to move through the obstacles, which is not the case. The obstacles in the figures are shown at the initial position. A similar plot for the obstacles with $2m$ safety distance can be seen in Figure 5.4. Here it can clearly be seen that the robot avoids hitting the obstacles, and the best trajectory, which is shown in green avoids the obstacles with a good margin.



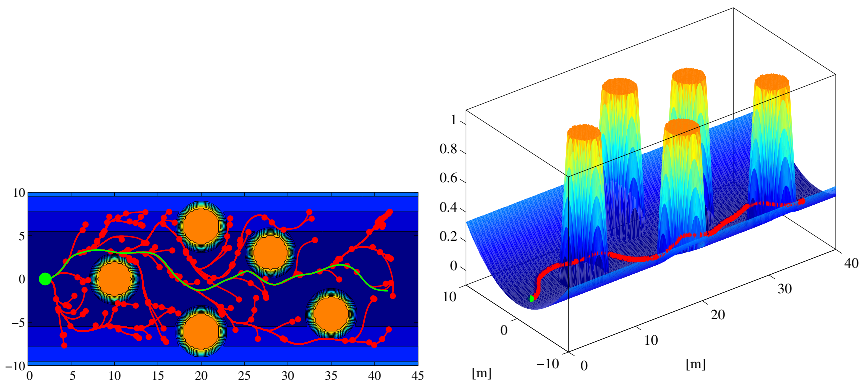
(a) A 2D plot of the potential field. The best trajectory is marked with green.

(b) A 3D plot of the potential field.

Figure 5.3: The RRT algorithm applied as a pure obstacle avoidance algorithm. In this simulation i the obstacle has a safety radius of $1m$ and are moving.

It is according to [Nowak et al., 2010] difficult to compare obstacle avoidance techniques to each other, and no benchmarks really exist. However, they mention different performance criteria for a successful obstacle avoidance algorithm. These are:

- Mission success
- Path length
- Time taken



(a) A 2D plot of the potential field. The best trajectory is marked with green.

(b) A 3D plot of the potential field.

Figure 5.4: The RRT algorithm applied as a pure obstacle avoidance algorithm. In this simulation i the obstacle has a safety radius of $2m$ and are static.

- Number of collisions
- Obstacle clearance
- Robustness in narrow spaces

According to these benchmarks our algorithm has mission success, since it is able to move forwards through a crowded environment. Not many pure obstacle avoidance techniques are, however, able to plan this far without an overall planner. Looking at the trajectories, the paths are relatively straight and thus cannot be much shorter, and thus the time taken is relatively short as well. Compared to for example Vector Field Histogram and Nearness Diagram, which have a tendency to slow down the robot, this is good. The algorithm avoids collisions and keep a good obstacle clearance, so this adds to the performance of the algorithm. However, the robustness for narrow places have not been tested, but it is assumed that it will perform well. According to these benchmarks, the developed RRT algorithm performs well as a pure obstacle avoidance technique.

## 5.6 Nearest Neighbour Search

Lastly an algorithm for increasing the speed of the nearest neighbour search, has been presented. Nearest neighbour search is a computational hard problem. It has an inevitable exponential growth relative to the dimension - both in theory and in practice[Yershova and LaValle, 2007]. In fact, the naive brute force approach will perform the best as the number of dimensions grows towards infinity. However, when working with the dimensions that are typical in planning algorithms, better algorithms can be found. A widely used algorithm for fast locating the nearest neighbour, is the $kd$-tree (see [LaValle, 2006, Choset et al., 2005]), which was developed in the field of computational geometry, and thus not for planning algorithms. Therefore it has some shortcomings, for which our algorithm works better. For example, a $kd$-tree works well for clustered points, but in the case of planning, it is attempted to avoid clustering to explore the configuration space well. A grid based method, like the one developed, splits the configuration space evenly, and is thus better for non clustered points.

A comparison of the performance of our developed box based algorithm to a brute force approach and to a $kd$-tree approach, is shown in Figure 5.5 for finding the nearest neighbour of 10000 vertices in two dimensions. For comparison to the $kd$-tree, an existing MATLAB implementation of the algorithm has been used [Tagliasacchi, 2010]. The performance of the $kd$-tree implementation in Figure 5.5 is seen to be worse than

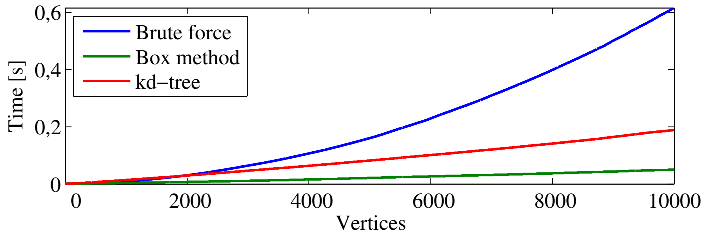


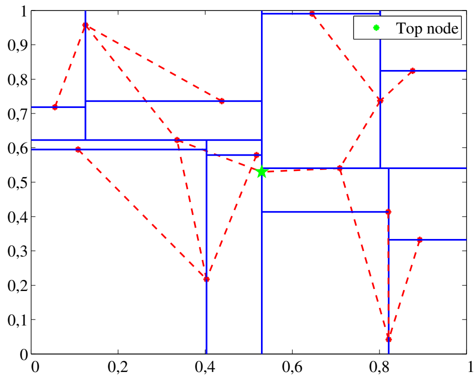
Figure 5.5: Comparison of the time it takes to find the nearest neighbour for two dimensions using the $kd$-tree as well as the brute force method and the box method.

using the box based method. Furthermore, the complexity of the $kd$-tree search algorithm is $\mathcal{O}(n^{1-\frac{1}{d}})$, where $n$ is the number of vertices and $d$ is the number of dimensions. This means that the $kd$-tree performs worse in higher dimensions, whereas the box based method performs better for e.g. four dimensions, which can be seen in *Paper E*. The speed of the $kd$-tree algorithm can, however, be improved by using approximate nearest neighbour (ANN) algorithms. This comes at a trade-off for not being entirely sure that it is the correct nearest neighbour, but an approximate. The bound on the complexity for ANN is $\mathcal{O}(\log n)$ [Karaman and Frazzoli, 2010, Yershova and LaValle, 2007], which is sill not better than the proposed method, which have a theoretical approximate performance of $\mathcal{O}(1)$, when the number of vertices to be added is known in advance.

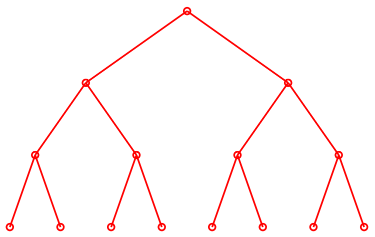
Another issue with the $kd$-tree, is that it is developed for static databases, and not expanding, like in a planning tree. This means, that if a $kd$-tree is built incrementally, it can get unbalanced and thus degrade in search performance. The implementation of the $kd$-tree in above experiment is using a balanced tree, and thus actually giving the $kd$-tree an advantage. But even though that is the case, our algorithm is still faster. In *Paper E* there was not enough space to explain why the $kd$-tree performance degrades, when it is unbalanced. This is explained in the following section.

### 5.6.1 A Balanced $kd$-tree

In *Paper E* it is noted that searching a balanced $kd$-tree is much more efficient than an unbalanced tree. This section shows why this is the case. When the $kd$-tree is built after all the vertices are known, it ensures equal depth of the whole tree. This means that there will be $2^{d-1}$ vertices in the level depth $d$, and there will be a maximum depth for a given number of vertices. An example of this is shown in Figure 5.6, where a $kd$-tree is built from 15 randomly selected vertices. An incrementally built $kd$-tree will however



(a) The graph of the $kd$-tree. The vertex corresponding the top node of the tree, is marked with a star.

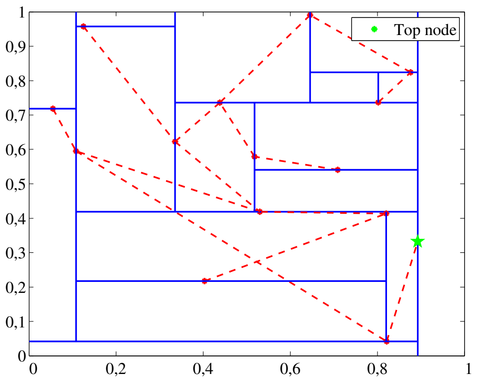


(b) The corresponding tree, with a height of 3.

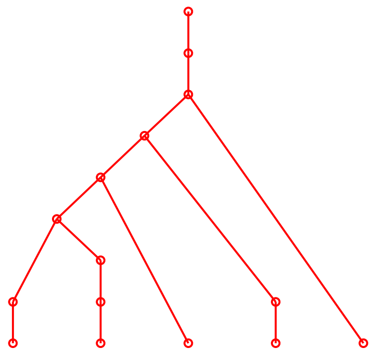
Figure 5.6: An example of a balanced $kd$-tree with 15 vertices.

not have this property, and depends on where the vertices are placed in the configuration space. Specifically if initial vertices are placed near the edges, it can increase the depth of the tree. To illustrate this, a tree with the same 15 points, as in Figure 5.6, is built incrementally. This is shown in Figure 5.7. First note that the vertices in the graph in Figure 5.7a are much longer and crosses each other much more compared to Figure 5.6a. Additionally the top and the next node of the tree only have one child in Figure 5.7b. This is because they were close to the edge. Comparing the resulting tree depth, it is seen that the balanced tree has a tree depth of only three, whereas the unbalanced tree has a depth of 8.

To illustrate this problem further, a $kd$-tree with 10000 vertices have been built. This is done with both a balanced and an unbalanced tree. A histogram of the depth of the vertices can be seen in Figure 5.8. It is seen that the unbalanced tree have an average depth of around 16 and a maximum around 35. This is in contrast to the maximum depth of 13 for the balanced tree. Such a large difference in tree depth, can impact the performance of the algorithm greatly.



(a) The graph of the $kd$-tree. The vertex corresponding the top node of the tree, is marked with a star.



(b) The corresponding tree, with a height of 8.

Figure 5.7: An example of an unbalanced $kd$-tree with 15 vertices. Because the first vertices are not close to the center, the graph becomes unbalanced.

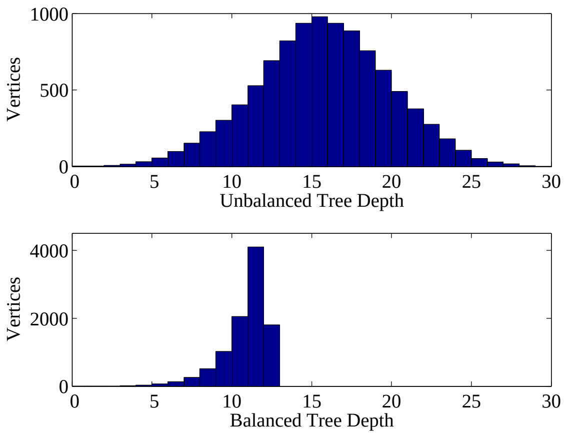


Figure 5.8: Histogram of the depth of the $kd$-tree with a balanced version and a non balanced version.

# 6 Conclusion

The work documented in this thesis treats the task of planning motion for a robot that drives around in a human environment. Applications of motion planning in human environments are numerous in a future world, where intelligent mobile robots have a potential to become an ubiquitous part of our everyday life.

There are many steps towards making a robot navigate safe and naturally around in a human environment, while at the same time performing a desired task. The underlying basis for this work, is a service robot driving around in a populated human environment. The robot must identify people that needs help and drive safe and natural through the environment. But at the same time, the motion algorithms must be usable in other robot applications in human environments.

The work in this thesis deals with this problem in the following steps. First, a necessary ability for a robot in a human environment is to be able to detect the pose of persons in the area. Next task is deciphering the human motion in relation to determining their interest in close interaction with the robot. This estimated interest in interaction is used to adapt the motion of the robot around the humans. To demonstrate the general application of this work, the system is expanded to a robot game, which have the purpose of motivating elderly people to exercise. The motion strategy around a person is then adapted to encompass environments with more people. This makes a robot capable of navigating safe and naturally through a human environment like a pedestrian street, a hospital, or an airport. The work with finding out how to navigate through a human crowd, inspired work on a new method for minimizing the time complexity for a general RRT planning algorithm, which ultimately improves the real time motion planning algorithm.

The contributions arising from the different tasks are documented in five papers, which are appended after this conclusion. In summary, the major contributions from the different papers can be listed as:

- An algorithm for inferring the pose of a person from laser range scanner measurements, is developed. The algorithm finds the persons in the laser range scan and feeds the continuous position updates through a measurement driven Kalman filter, which provides position and velocity. To obtain the pose, the velocity estimate is post filtered in an autoregressive filter.

- An algorithm that estimates a person's interest in interaction with the robot. The interest in interaction is estimated using a Case-Based Reasoning system, which learns from previous experiences to gain knowledge about how persons behave when they are interested or not interested in close interaction.

- A human-aware motion planner, that adjust the robot motion according to the situation. The motion planner uses the perceived interest in interaction to generate a potential field around the person. To move appropriately the robot uses the gradient of the potential field to move towards the lowest point. As the perceived interest level changes over time, the potential field, and thus the robot motion, adapts continuously to the changed situation.

- A successful integration of these three algorithms on a robotic system, which is demonstrated to be able to find out if a person is interested in interaction, and move according to the perceived interest.

- The general applicability of the proposed framework, is demonstrated by modifying it to be a robot game, where a robot is playing with a person. The game framework resembles a simple game of tag, where a person has to hand over an object to the robot. The better the player is, the harder the robot makes the task. The game is intended for maintaining and improving the physical health state of elderly people by encouraging them to exercise.

- The developed potential field, for the human-aware motion planner, is modified to support navigation in an area with many people. The trajectory planning problem is then formulated as an optimisation problem, where the cost of traversing the trajectory in the potential field should be minimised.

- The trajectory planning problem is solved by modifying an RRT algorithm. The RRT algorithm is modified to enable the costs of traversing the potential field to be minimized and to focus on keeping a desired pace instead of reaching a goal. Further, a new control input sampling strategy that utilizes a second order robot motion model, is presented. Last an MPC scheme is used to enable the RRT planner to continuously plan a reachable and feasible trajectory in an on-line system.

- An algorithm that minimises the computational cost of the RRT, by minimising the cost of searching for the nearest neighbour, algorithm is finally developed. It is shown that the algorithm obtains faster and better trajectories compared to existing approaches for fast nearest neighbour searches.

## 6.1 Future Work and New Ideas

There are several ways the work in this thesis can be extended, and many challenges for robot motion in human environments still exist. A few examples of potential further work is presented here.

- A straightforward extension to the work with the robot game, is to actually experiment with the game to study how it works in a real world nursery home. This have already been done, but the results have not been published yet. However, a video demonstrating the experiment was presented at the HRI2011 conference [Hansen, 2011]. Two snapshots from this video are shown in Figure 6.1.
- Currently the game is a single player game, but in a real world nursery home, it would be interesting with a multiplayer game, where more people could play at the same time.
- The complete trajectory planning algorithm has been implemented as a Player/Stage plugin. This should also make it straightforward to test this in a real world scenario.
- In the trajectory planning problem, the persons are treated as moving individually around. But in a real world, people are also moving together or standing and talking in groups. It would be interesting to try to estimate this human motion correlation and use this information to improve the navigation algorithm.
- In the trajectory planning algorithm, vertices with a significant number of failed expansions are most likely close to obstacles or people. Therefore a strategy to prune those vertices can be implemented.
- The human-aware motion planner adapts the robot motion according to one specific person, which might be interested in interaction. The trajectory planning algorithm navigates around, treating all humans as if they were not interested in interaction. But imagine a service robot that has to determine which person needs most help, and hence which person to start interaction with. So an algorithm is needed to make the robot figure out if it should switch motion strategy from navigation through the environment to motion planning around a specific person. This means that the robot should evaluate if the reward associated with focusing on one specific person, over weighs the cost of doing it. Furthermore the robot may also affect the intention of the person by the choices it makes. A possible solution scheme to deal with this type of problem, might be a simple Partially Observable Markov Decision Process (POMDP).

(a) (b)

Figure 6.1: Two snapshots from the video [Hansen, 2011], where the robot game is played on a nursery home.

### 6.1.1 POMPD's for Determining Actions

This section briefly discusses the above mentioned new idea for a solution scheme, where POMDP's are used to determine if a robot should start approaching a person, or continue moving around. In the POMDP context, the robot should evaluate if the reward associated with focusing on one specific person, over weighs the cost of doing so. Furthermore the robot may also affect the intention of the person by the choice it makes. A simple POMDP model for a single person is illustrated in Figure 6.2. $X1$ and $X2$ are the two different

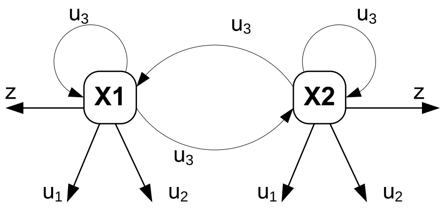


Figure 6.2: Illustration of a POMDP framework to find out how the robot should behave. $X_1$ and $X_2$ are the states a person can be in (interested or not interested in interaction). $u_1, u_2, u_3$ are the actions the robot can take (start interaction, disregard person, observe further). $z$ is the estimated probability of the person being in the state.

states a person can be in. Either the person wants to interact with the robot ($X1$), or the person is not interested ($X2$). $z =$*PI* is the estimate of the person interest in interaction, which gives an indication to which state the person is in. So if for example $z = 0.7$, it can be interpreted as there is $70\%$ chance that the person is in state $X1$, and a $30\%$ chance that the person is in state $X2$. $u_1, u_2, u_3$ are the different actions the robot can take. $u_1$ is to start interacting with the robot, $u_2$ is to disregard the person and $u_3$ is to further observe the person to improve the estimate of *PI* ($z$ gets closer to 0 or 1). However, as seen in Figure 6.2, there is a possibility that the person changes state if the robot continues to observe the person. For example the person might start to get interested if the robot continues to hang around. This probability is denoted $P(X1'|X2, u_3)$. For each of the actions there is an associated reward function that gives a payoff or a cost. Examples of rewards and state transition probabilities can be:

$$\begin{aligned} r(u_1|X1) = 100 \qquad & r(u_1|X2) = -50 \\ r(u_2|X1) = -80 \qquad & r(u_2|X2) = 50 \\ r(u_3|X1) = -5 \qquad & r(u_3|X2) = -5 \\ P(X1'|X1, u_3) \quad = \quad & 1 - P(X2'|X1, u_3) = 0.05 \\ P(X1'|X2, u_3) \quad = \quad & 1 - P(X2'|X2, u_3) = 0.1 \end{aligned} \tag{6.1}$$

If the person is interested and $u_1$ is chosen, there is a positive reward. However, if the person is interested and $u_2$ is chosen it is not so good and it has a negative cost. Choosing $u_3$ also has a mild cost, because the robot might not find other persons, who needs help. Using this framework, POMDP solution methods can be applied to find the optimal strategy.

# References

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

[Agger, 2009] Agger, P. (2009). Sociale robotter. The Danish Etichal Council: http://www.etiskraad.dk/graphics/03_udgivelser/publikationer/ovrige_udgivelser/sociale-robotter.pdf.

[Aha, 1998] Aha, D. (1998). The omnipresence of case-based reasoning in science and application. *Knowledge-Based Systems*, 11(5-6):261–273.

[Alpaydin, 2004] Alpaydin, E. (2004). *Introduction to machine learning*. The MIT Press.

[Althaus et al., 2004] Althaus, P., Ishiguro, H., Kanda, T., Miyashita, T., and Christensen, H. (2004). Navigation for human-robot interaction tasks. In *Proceedings of IEEE International Conference on Robotics and Automation, 2004. ICRA '04.*, volume 2, pages 1894–1900.

[Andersen et al., 2008] Andersen, H. J., Bak, T., and Svenstrup, M. (2008). Adaptive robot to person encounter. In *Proceedings of the International Conference on Research and Education in Robotics - EUROBOT 2008*, Lecture Notes in Computer Science, pages 13–23, Heidelberg, Germany. MATFYZPRESS.

[Argyros et al., 2002] Argyros, A., Georgiadis, P., Trahanias, P., and Tsakiris, D. (2002). Semi-autonomous navigation of a robotic wheelchair. *J. Intell. Robotics Syst.*, 34:315–329.

[Arkin, 2008] Arkin, R. C. (2008). Governing lethal behavior: embedding ethics in a hybrid deliberative/reactive robot architecture. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 121–128, New York, NY, USA. ACM.

[Arras et al., 2007] Arras, K., Mozos, O., and Burgard, W. (2007). Using boosted features for the detection of people in 2d range data. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3402 –3407.

[Asimov, 1968] Asimov, I. (1968). Runaround. In *I, Robot (a collection of short stories originally published between 1940 and 1950)*, pages 33–51. Grafton Books, London.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagrams -a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23:345–405.

[Bak et al., 2001] Bak, M., Poulsen, N. K., and Ravn, O. (2001). Path following mobile robot in the presence of velocity constraints. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

[Barraquand et al., 1992] Barraquand, J., Langlois, B., and Latombe, J.-C. (1992). Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224 –241.

[Bauer et al., 2009] Bauer, A., Klasing, K., Lidoris, G., Mühlbauer, Q., Rohrmüller, F., Sosnowski, S., Xu, T., Kühnlenz, K., Wollherr, D., and Buss, M. (2009). The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 1(2):127–140.

[Bellotto and Hu, 2009] Bellotto, N. and Hu, H. (2009). Multisensor-based human detection and tracking for mobile service robots. *Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 39(1):167–181. PT: J; NR: 31; TC: 0; J9: IEEE TRANS SYST MAN CYBERN B; PG: 15; GA: 395ZT.

[Bennewitz, 2004] Bennewitz, M. (2004). *Mobile Robot Navigation in Dynamic Environments*. PhD thesis, University of Freiburg, Department of Computer Science.

[Bennewitz et al., 2005] Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31.

[Bennewitz et al., 2002] Bennewitz, M., Burgard, W., and Thrun, S. (2002). Learning motion patterns of persons for mobile service robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 4, pages 3601 – 3606 vol.4.

[Betts, 1998] Betts, J. T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control and Dynamics*, 21(2):193–207.

[Borenstein and Koren, 1991] Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278 –288.

[Breazeal, 2002] Breazeal, C. (2002). *Designing Sociable Robots*. MIT Press, Cambridge, MA, USA.

[Brick and Scheutz, 2007] Brick, T. and Scheutz, M. (2007). Incremental natural language processing for hri. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, HRI '07, pages 263–270, New York, NY, USA. ACM.

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

[Brooks et al., 2009] Brooks, A., Kaupp, T., and Makarenko, A. (2009). Randomised mpc-based motion-planning for mobile robot obstacle avoidance. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3962–3967.

[Bruce and Gordon, 2004] Bruce, A. and Gordon, G. (2004). Better motion prediction for people-tracking. In *Robotics and Automation, 2004. ICRA '04. IEEE International Conference on*.

[Burgard et al., 1998] Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1998). The interactive museum tour-guide robot. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 11–18, Menlo Park, CA, USA. American Association for Artificial Intelligence.

[Calinon and Billard, 2007] Calinon, S. and Billard, A. (2007). What is the teacher's role in robot programming by demonstration? - toward benchmarks for improved learning. *Interaction Studies. Special Issue on Psychological Benchmarks in Human-Robot Interaction.*, 8(3):441–464.

[Chazelle, 1987] Chazelle, B. (1987). Approximation and decomposition of shapes. In Schwartz, J. T. and Yap, C. K., editors, *Algorithmic and Geometric Aspects of Robotics*, pages 145–185. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Choset et al., 2005] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA.

[Cielniak et al., 2005] Cielniak, G., Treptow, A., and Duckett, T. (2005). Quantitative performance evaluation of a people tracking system on a mobile robot. In *Proc. 2nd European Conference on Mobile Robots*.

[COGNIRON, 2007] COGNIRON (2007). http://www.cogniron.org/.

[Collett et al., 2005] Collett, T., MacDonald, B. A., and Gerkey, B. P. (2005). Player 2.0: Toward a practical robot programming framework. In Sammut, C., editor, *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, Sydney, Australia. http://playerstage.sourceforge.net.

[COSY, 2007] COSY (2007). http://www.cognitivesystems.org/.

[Dautenhahn, 2007] Dautenhahn, K. (2007). Methodology & themes of human-robot interaction: A growing research field. *International Journal of Advanced Robotic Systems*, 4(1):103–108.

[Dautenhahn et al., 2006] Dautenhahn, K., Walters, M., Woods, S., Koay, K. L., Sisbot, E. A., Alami, R., and Siméon, T. (2006). How may i serve you? a robot companion approaching a seated person in a helping context. In *HRI Human Robot Interaction '06 (HRI06)*, Salt Lake City, Utah, USA.

[de Berg et al., 2000] de Berg, M., van Krefeld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications, Second Edition*. Springer, 2nd edition.

[Dementia, 2010] Dementia (2010). Dementia. http://www.videnscenterfordemens.dk. Danish Dementia Research Center.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[Dornaika and Raducanu, 2008] Dornaika, F. and Raducanu, B. (2008). Detecting and tracking of 3d face pose for human-robot interaction. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1716–1721.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *Robotics Automation Magazine, IEEE*, 13(2):99 –110.

[Edelsbrunner, 1987] Edelsbrunner, H. (1987). *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin.

[Fajen and Warren, 2003] Fajen, B. and Warren, W. (2003). The behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology: Human Perception and Performance*, 29(1):343–362.

[Fajen et al., 2003] Fajen, B., Warren, W., Temizer, S., and Kaelbling, L. (2003). A dynamical model of visually-guided steering, obstacle avoidance, and route selection. *International Journal of Computer Vision*, 54(1):1–3.

[Feil-Seifer and Matarić, 2005] Feil-Seifer, D. J. and Matarić, M. J. (2005). A multi-modal approach to selective interaction in assistive domains. In *IEEE Proceedings of the International Workshop on Robot and Human Interactive Communication*, pages 416–421, Nashville, TN.

[Ferguson et al., 2006] Ferguson, D., Kalra, N., and Stentz, A. (2006). Replanning with rrts. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2006*, pages 1243–1248.

[Ferguson and Stentz, 2006] Ferguson, D. and Stentz, A. (2006). Anytime rrts. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5369–5375.

[Ferguson and Stentz, 2007] Ferguson, D. and Stentz, A. (2007). Anytime, dynamic planning in high-dimensional search spaces. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1310–1315.

[Fikes and Nilsson, 1971] Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

[Fiorini and Shiller, 1998] Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760.

[Fod et al., 2002] Fod, A., Howard, A., and Mataric, M. J. (2002). Laser-based people tracking. In *In Proc. of the IEEE International Conference on Robotics & Automation (ICRA*, pages 3024–3029.

[Foka, 2005] Foka, A. (2005). *Predictive Autonomous Robot Navigation*. PhD thesis, Department of Computer Science, University of Crete, Greece.

[Foka and Trahanias, 2007] Foka, A. and Trahanias, P. (2007). Real-time hierarchical pomdps for autonomous robot navigation. *Robot. Auton. Syst.*, 55:561–571.

[Foka and Trahanias, 2010] Foka, A. and Trahanias, P. (2010). Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *International Journal of Social Robotics*, 2(1):79–94.

[Fong et al., 2003] Fong, T. W., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42:143–166.

[Fox et al., 1997] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23 –33.

[Gates, 2007] Gates, B. (2007). A robot in every home. *Scientific American*, 296(1):58–65.

[Giusti and Marti, 2006] Giusti, L. and Marti, P. (2006). Interpretative dynamics in human robot interaction. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006.*, pages 111–116.

[Glas et al., 2007] Glas, D., Miyashita, T., Ishiguro, H., and Hagita, N. (2007). Laser tracking of human body motion using adaptive shape modeling. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 602–608.

[Gockley et al., 2007] Gockley, R., Forlizzi, J., and Simmons, R. (2007). Natural person-following behavior for social robots. In *Proceeding of the ACM/IEEE international conference on Human-robot interaction, HRI '07*, pages 17–24, New York, NY, USA. ACM.

[Goodrich and Schultz, 2007] Goodrich, M. and Schultz, A. (2007). Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275.

[Govea, 2007] Govea, D. A. V. (2007). *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*. PhD thesis, Institut National Polytechnique de Grenoble.

[Gulati and Kuipers, 2008] Gulati, S. and Kuipers, B. (2008). High performance control for graceful motion of an intelligent wheelchair. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3932 –3938.

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

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

[Hansen, 2011] Hansen, S. T. (2011). Robot games for elderly. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 413–414, New York, NY, USA. ACM.

[Hansen et al., 2009a] Hansen, S. T., Svenstrup, M., Andersen, H. J., and Bak, T. (2009a). Adaptive human aware navigation based on motion pattern analysis. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages –, Toyama, Japan.

[Hansen et al., 2009b] Hansen, S. T., Svenstrup, M., Andersen, H. J., Bak, T., and Jensen, O. B. (2009b). The santabot experiment: a pilot study of human-robot interaction.

Video for the 4th ACM/IEEE international conference on Human robot interaction (HRI) video session.

[Hansen et al., 2009c] Hansen, S. T., Svenstrup, M., Andersen, H. J., Bak, T., and Jensen, O. B. (2009c). The santabot experiment: a pilot study of human-robot interaction. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 211–212, New York, NY, USA. ACM.

[Hansen et al., 2010] Hansen, S. T., Svenstrup, M., and Dalgaard, L. (2010). An adaptive robot game. In *ISR/ROBOTIK 2010 (Proceedings of the joint conference of ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics))*, pages 76–83. VDE Verlag.

[Hart et al., 1968] Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100 –107.

[Hoeller et al., 2007] Hoeller, F., Schulz, D., Moors, M., and Schneider, F. (2007). Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1260 –1265.

[Jenkins et al., 2007] Jenkins, O. C., Serrano, G. G., and Loper, M. M. (2007). *Recognizing Human Pose and Actions for Interactive Robots*, chapter 6, pages 119–138. I-Tech Education and Publishing.

[Jun et al., 1999] Jun, M., Roumeliotis, S., and Sukhatme, G. (1999). State estimation of an autonomous helicopter using kalman filtering. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1346–1353vol.3.

[Jurisica and Glasgow, 1995] Jurisica, I. and Glasgow, J. (1995). Applying case-based reasoning to control in robotics. In *3rd Robotics and Knowledge-Based Systems Workshop, St. Hubert Quebec*.

[Kanda et al., 2007] Kanda, T., Sato, R., Saiwaki, N., and Ishiguro, H. (2007). A two-month field trial in an elementary school for long-term human-robot interaction. *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 23(5):962–971.

[Karaman and Frazzoli, 2010] Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain.

[Kavraki et al., 1996] Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566 –580.

[Kelley et al., 2008] Kelley, R., Tavakkoli, A., King, C., Nicolescu, M., Nicolescu, M., and Bebis, G. (2008). Understanding human intentions via hidden markov models in autonomous mobile robots. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 367–374, New York, NY, USA. ACM.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90.

[Kirby et al., 2009] Kirby, R., Simmons, R., and Forlizzi, J. (2009). Companion: A constraint-optimizing method for person-acceptable navigation. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 607 –612.

[Kleinehagenbrock et al., 2002] Kleinehagenbrock, M., Lang, S., Fritsch, J., Lomker, F., Fink, G., and Sagerer, G. (2002). Person tracking with a mobile robot based on multimodal anchoring. *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN), 2002.*, 1:423–429.

[Kluge et al., 2001] Kluge, B., Kohler, C., and Prassler, E. (2001). Fast and robust tracking of multiple moving objects with a laser range finder. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1683 – 1688 vol.2.

[Koay et al., 2007] Koay, K. L., Sisbot, E. A., Syrdal, D. A., Walters, M. L., Dautenhahn, K., and Alami, R. (2007). Exploratory study of a robot approaching a person in the context of handling over an object. In *Association for the Advancement of Artificial Intelligence Spring Symposia, AAAI*, Palo Alto, CA, USA.

[Koay et al., 2005] Koay, K. L., Walters, M., and Dautenhahn, K. (2005). Methodological issues using a comfort level device in human-robot interactions. In *IEEE International Workshop on Robot and Human Interactive Communication, 2005. ROMAN 2005.*, pages 359–364.

[Kolodner, 1993] Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Kondaxakis et al., 2009] Kondaxakis, P., Baltzakis, H., and Trahanias, P. (2009). Learning moving objects in a multi-target tracking scenario for mobile robots that use laser range measurements. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 1667–1672, Piscataway, NJ, USA. IEEE Press.

[Kracht and Nielsen, 2007] Kracht, S. and Nielsen, C. (2007). Robots in everyday human environments. Master's thesis, Aalborg University.

[Kulic and Croft, 2007] Kulic, D. and Croft, E. (2007). Affective state estimation for human-robot interaction. *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 23(5):991–1000.

[Lam et al., 2010] Lam, C.-P., Chou, C.-T., Chang, C.-F., and Fu, L.-C. (2010). Human-centered robot navigation – toward a harmoniously coexisting multi-human and multi-robot environment. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1813 –1818.

[Lam et al., 2011] Lam, C.-P., Chou, C.-T., Chiang, K.-H., and Fu, L.-C. (2011). Human-centered robot navigation - towards a harmoniously human – robot coexisting environment. *Robotics, IEEE Transactions on*, 27(1):99 –112.

[LaValle, 2006] LaValle, S. (2006). *Planning algorithms*. Cambridge Univ Pr.

[LaValle and Kuffner Jr, 2001] LaValle, S. and Kuffner Jr, J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378.

[Likhachev and Arkin, 2001] Likhachev, M. and Arkin, R. (2001). Spatio-temporal case-based reasoning for behavioral selection. In *Proc. IEEE International Conference on Robotics and Automation, ICRA*, volume 2, pages 1627–1634.

[Lindemann and LaValle, 2005] Lindemann, S. R. and LaValle, S. M. (2005). Current issues in sampling-based motion planning. In Dario, P. and Chatila, R., editors, *Robotics Research*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 36–54. Springer Berlin / Heidelberg.

[Lu and Milios, 1994] Lu, F. and Milios, E. (1994). Robot pose estimation in unknown environments by matching 2d range scans. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 935 –938.

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA.

[Martinez-Garcia et al., 2005] Martinez-Garcia, E., Akihisa, O., and Yuta, S. (2005). Crowding and guiding groups of humans by teams of mobile robots. In *Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on*, pages 91 – 96.

[Michalowski et al., 2007] Michalowski, M. P., Sabanovic, S., DiSalvo, C. F., Font, D. B., Hiatt, L. M., Melchior, N., and Simmons, R. (2007). Socially distributed perception: Grace plays social tag at aaai 2005. *Autonomous Robots*, 22(4):385–397.

[Michalowski and Simmons, 2006] Michalowski, M. P. and Simmons, R. (2006). Multimodal person tracking and attention classification. In *Proceedings of the 1st Annual Conference on Human-Robot Interaction (HRI 2006)*, pages 347–348. ACM.

[Minguez and Montano, 2004] Minguez, J. and Montano, L. (2004). Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios. *Robotics and Automation, IEEE Transactions on*, 20(1):45 – 59.

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., and Whittaker, W. (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 695 – 701 vol.1.

[Müller et al., 2008] Müller, J., Stachniss, C., Arras, K., and Burgard, W. (2008). Socially inspired motion planning for mobile robots in populated environments. In *In Proceedings of the International Conference on Cognitive Systems (CogSy)*.

[Mumm and Mutlu, 2011] Mumm, J. and Mutlu, B. (2011). Human-robot proxemics: physical and psychological distancing in human-robot interaction. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 331–338, New York, NY, USA. ACM.

[Munoz-Salinas et al., 2005] Munoz-Salinas, R., Aguirre, E., Garcia-Silvente, M., and Gonzalez, A. (2005). People detection and tracking through stereo vision for human-robot interaction. *MICAI 2005: Advances in Artificial Intelligence*, 3789/2005:337–346.

[Mutlu and Forlizzi, 2008] Mutlu, B. and Forlizzi, J. (2008). Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 287–294, New York, NY, USA. ACM.

[Nabe et al., 2006] Nabe, S., Kanda, T., Hiraki, K., Ishiguro, H., Kogure, K., and Hagita, N. (2006). Analysis of human behavior to a communication robot in an open field. In *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 234–241, New York, NY, USA. ACM.

[Nakauchi and Simmons, 2000] Nakauchi, Y. and Simmons, R. (2000). A social robot that stands in line. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 357 –364 vol.1.

[Nowak et al., 2010] Nowak, W., Zakharov, A., Blumenthal, S., and Prassler, E. (2010). Benchmarks for mobile manipulation and robust obstacle avoidance and navigation. Technical report, GPS Gesellschaft für Produktionssysteme GmbH.

[Pacchierotti et al., 2006a] Pacchierotti, E., Christensen, H., and Jensfelt, P. (2006a). Evaluation of passing distance for social robots. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 315–320.

[Pacchierotti et al., 2006b] Pacchierotti, E., Christensen, H., and Jensfelt, P. (2006b). Evaluation of passing distance for social robots. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 315 –320.

[Pacchierotti et al., 2005] Pacchierotti, E., Christensen, H. I., and Jensfelt, P. (2005). Embodied social interaction for service robots in hallway environments. In *Proc. of the International Conference on Field and Service Robotics (FSR'05)*.

[Pateraki et al., 2009] Pateraki, M., Baltzakis, H., Kondaxakis, P., and Trahanias, P. (2009). Tracking of facial features to support human-robot interaction. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3755 – 3760.

[Pearl, 1984] Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Prado, 2007] Prado, M. (2007). Optimal velocity planning of wheeled mobile robots on specific paths in static and dynamic environments. In Kolski, S., editor, *Mobile Robots: Perception & Navigation*, pages 357–382. INTECH.

[Prassler et al., 2002] Prassler, E., Bank, D., and Kluge, B. (2002). Motion coordination between a human and a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and System, 2002.*, volume 2, pages 1228–1233.

[Pratichizzo et al., 2010] Pratichizzo, D., Malvezzi, M., and Bicchi, A. (2010). On motion and force controllability of grasping hands with postural synergies. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain.

[Ram et al., 1997] Ram, A., Arkin, R. C., Moorman, K., and Clark, R. J. (1997). Case-based reactive navigation: A case-based method for on-line selection and adaptation of reactive control parameters in autonomous robotic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 27:376–394.

[Rodgers et al., 2006] Rodgers, J., Anguelov, D., Pang, H.-C., and Koller, D. (2006). Object pose detection in range scan data. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2:2445–2452.

[Sasaki and Hashimoto, 2006] Sasaki, T. and Hashimoto, H. (2006). Human observation based mobile robot navigation in intelligent space. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1044 –1049.

[Satake et al., 2009] Satake, S., Kanda, T., Glas, D. F., Imai, M., Ishiguro, H., and Hagita, N. (2009). How to approach humans?: strategies for social robots to initiate interaction. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, HRI '09, pages 109–116, New York, NY, USA. ACM.

[Schmidt-Rohr et al., 2008] Schmidt-Rohr, S. R., Knoop, S., Lösch, M., and Dillmann, R. (2008). Reasoning for a multi-modal service robot considering uncertainty in human-robot interaction. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 249–254, New York, NY, USA. ACM.

[Schulz et al., 2001] Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2001). Tracking multiple moving objects with a mobile robot. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, 1:371–377.

[Shi et al., 2008] Shi, D., Collins, E., Donate, A., Liu, X., Goldiez, B., and Dunlap, D. (2008). Human-aware robot motion planning with velocity constraints. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pages 490 –497.

[Shiomi et al., 2008] Shiomi, M., Sakamoto, D., Takayuki, K., Ishi, C. T., Ishiguro, H., and Hagita, N. (2008). A semi-autonomous communication robot: a field trial at a train station. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 303–310, New York, NY, USA. ACM.

[Shkolnik et al., 2009] Shkolnik, A., Walter, M., and Tedrake, R. (2009). Reachability-guided sampling for planning under differential constraints. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2859–2865.

[Siciliano and Khatib, 2008] Siciliano, B. and Khatib, O. (2008). *Handbook of Robotics*. Springer-Verlag, Heidelberg.

[Sisbot et al., 2005] Sisbot, E., Alami, R., Simeon, T., Dautenhahn, K., Walters, M., and Woods, S. (2005). Navigation in the presence of humans. In *5th IEEE-RAS international Conference on Humanoid Robots, 2005*, pages 181–188.

[Sisbot et al., 2006] Sisbot, E., Clodic, A., Marin U., L., Fontmarty, M., Brethes, L., and Alami, R. (2006). Implementing a human-aware robot system. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006.*, pages 727–732.

[Sisbot, 2008] Sisbot, E. A. (2008). *Towards Human-Aware Robot Motion*. PhD thesis, LAAS-CNRS, Tolouse.

[Sisbot et al., 2008] Sisbot, E. A., Clodic, A., Alami, R., and Ransan, M. (2008). Supervision and motion planning for a mobile manipulator interacting with humans. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 327–334, New York, NY, USA. ACM.

[Smith et al., 1998] Smith, I., Cohen, P., Bradshaw, J., Greaves, M., and Holmback, H. (1998). Designing conversation policies using joint intention theory. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, ICMAS '98, pages 269–, Washington, DC, USA. IEEE Computer Society.

[Sørensen, 2010] Sørensen, L. V. (2010). Demens: Ikke-farmalogigske interventioner - en kommenteret udenlands medicinsk teknologivurdering. *Monitorering & Medicinsk Teknologivurdering*, 3:1–34.

[Stentz, 1994] Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310 –3317 vol.4.

[Stiefelhagen et al., 2007] Stiefelhagen, R., Ekenel, H., Fugen, C., Gieselmann, P., Holzapfel, H., Kraft, F., Nickel, K., Voit, M., and Waibel, A. (2007). Enabling multimodal human-robot interaction for the karlsruhe humanoid robot. *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 23(5):840–851.

[Svenstrup, 2010] Svenstrup, M. (2010). Sampling based trajectory planning for robots in dynamic human environments. In *Extended abstract in workshop material from Robotics Science and Systems (RSS) 2010.*

[Svenstrup et al., 2010a] Svenstrup, M., Bak, T., and Andersen, H. J. (2010a). Trajectory planning for robots in dynamic human environments. In *IROS 2010: The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan.

[Svenstrup et al., 2011] Svenstrup, M., Bak, T., and Andersen, H. J. (2011). Minimising computational complexity of the rrt algorithm - a practical approach. In *IEEE International Conference on Robotics and Automation, 2011. ICRA '11.*

[Svenstrup et al., 2008] Svenstrup, M., Bak, T., Maler, O., Andersen, H. J., and Jensen, O. B. (2008). Pilot study of person robot interaction in a public transit space. In *Proceedings of the International Conference on Research and Education in Robotics - EUROBOT 2008*, Lecture Notes in Computer Science, pages 120–131, Heidelberg, Germany. Springer-Verlag GmbH.

[Svenstrup et al., 2009] Svenstrup, M., Hansen, S. T., Andersen, H. J., and Bak, T. (2009). Pose estimation and adaptive robot behaviour for human-robot interaction. In *IEEE International Conference on Robotics and Automation, 2009. ICRA '09.*, pages 3571–3576, Kobe, Japan.

[Svenstrup et al., 2010b] Svenstrup, M., Hansen, S. T., Andersen, H. J., and Bak, T. (2010b). Adaptive human aware robot navigation in close proximity to humans. *Submitted to International Journal of Advanced Robotic Systems.*

[Sverre Syrdal et al., 2006] Sverre Syrdal, D., Dautenhahn, K., Woods, S., Walters, M., and Koay, K. L. (2006). Doing the right thing wrong - personality and tolerance to uncomfortable robot approaches. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006.*, pages 183–188.

[Tadokoro et al., 1995] Tadokoro, S., Hayashi, M., Manabe, Y., Nakami, Y., and Takamori, T. (1995). On motion planning of mobile robots which coexist and cooperate with human. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 2, pages 518 –523 vol.2.

[Tagliasacchi, 2010] Tagliasacchi, A. (2010). kd-tree for matlab. MATLAB Central File Exchange.

[Takayama and Pantofaru, 2009] Takayama, L. and Pantofaru, C. (2009). Influences on proxemic behaviors in human-robot interaction. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5495 –5502.

[Takemura et al., 2007] Takemura, H., Ito, K., and Mizoguchi, H. (2007). Person following mobile robot under varying illumination based on distance and color information. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, pages 1500 –1505.

[Tedrake, 2009] Tedrake, R. (2009). Lqr-trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems*, Seattle, USA.

[Thrun et al., 1999] Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). Minerva: a second-generation museum tour-guide robot. In *Robotics and Automation,* *1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 1999 –2005 vol.3.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

[Topp and Christensen, 2005] Topp, E. and Christensen, H. (2005). Tracking for following and passing persons. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2321 – 2327.

[Ulrich and Borenstein, 2000] Ulrich, I. and Borenstein, J. (2000). Vfh*: Local obstacle avoidance with look-ahead verification. In *IEEE International Conference on Robotics and Automation*, pages 2505–2511, San Francisco.

[van den Berg et al., 2008] van den Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928 –1935.

[van der Merwe et al., 2004] van der Merwe, R., Wan, E., Julier, S., Bogdanov, A., Harvey, G., and Hunt, J. (2004). Sigma-point kalman filters for nonlinear estimation and sensor fusion: Applications to integrated navigation. In *AIAA Guidance Navigation & Control Conference*.

[Vinciarelli et al., 2009] Vinciarelli, A., Pantic, M., and Bourlard, H. (2009). Social signal processing: Survey of an emerging domain. *Image and Vision Computing*, 27(12):1743–1759.

[Walters et al., 2005a] Walters, M. L., Dautenhahn, K., Koay, K. L., Kaouri, C., te Boekhorst, R., Nehaniv, C., Werry, I., and Lee, D. (2005a). Close encounters: Spatial distances between people and a robot of mechanistic appearance. In *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, pages 450–455, Tsukuba, Japan.

[Walters et al., 2005b] Walters, M. L., Dautenhahn, K., te Boekhorst, R., Koay, K. L., Kaouri, C., Woods, S., Nehaniv, C., Lee, D., and Werry, I. (2005b). The influence of subjects' personality traits on personal spatial zones in a human-robot interaction experiment. In *Proc. IEEE Ro-man*, pages 347–352, Hashville.

[Xavier et al., 2005] Xavier, J., Pacheco, M., Castro, D., Ruano, A., and Nunes, U. (2005). Fast line, arc/circle and leg detection from laser scan data in a player driver. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005.*, pages 3930–3935.

[Yamaoka et al., 2008] Yamaoka, F., Kanda, T., Ishiguro, H., and Hagita, N. (2008). How close?: model of proximity control for information-presenting robots. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, HRI ’08, pages 137–144, New York, NY, USA. ACM.

[Yershova and LaValle, 2007] Yershova, A. and LaValle, S. M. (2007). Improving motion-planning algorithms by efficient nearest-neighbor searching. *Robotics, IEEE Transactions on*, 23(1):151 –157.

[Yoda and Shiota, 1995] Yoda, M. and Shiota, Y. (1995). Basic study on avoidance motions for human behaviors. In *Robot and Human Communication, 1995. RO-MAN’95 TOKYO, Proceedings., 4th IEEE International Workshop on*, pages 318 –322.

[Yoda and Shiota, 1997] Yoda, M. and Shiota, Y. (1997). The mobile robot which passes a man. In *Robot and Human Communication, 1997. RO-MAN ’97. Proceedings., 6th IEEE International Workshop on*, pages 112 –117.

[Yoshimi et al., 2006] Yoshimi, T., Nishiyama, M., Sonoura, T., Nakamoto, H., Tokura, S., Sato, H., Ozaki, F., Matsuhira, N., and Mizoguchi, H. (2006). Development of a person following robot with vision based target detection. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5286 –5291.

[Zender et al., 2007] Zender, H., Jensfelt, P., and Kruijff, G.-J. M. (2007). Human- and situation-aware people following. In *Proc. of the 16th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2007)*, Jeju Island, Korea.

[Zhao and Shibasaki, 2005] Zhao, H. and Shibasaki, R. (2005). A novel system for tracking pedestrians using multiple single-row laser-range scanners. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(2):283 – 291.

[Zucker et al., 2007] Zucker, M., Kuffner, J., and Branicky, M. (2007). Multipartite rrts for rapid replanning in dynamic environments. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1603–1609.

# Contributions

# Paper A

**Pose Estimation and Adaptive Robot Behaviour for Human-Robot Interaction**

Mikael Svenstrup, Søren Tranberg Hansen, Hans Jørgen Andersen, Thomas Bak

This paper is published in:
Proceedings of the 2009 International Conference Robotics and Automation,
ICRA 2009

**Abstract**

This paper introduces a new method to determine a person's pose based on laser range measurements. Such estimates are typically a prerequisite for any human-aware robot navigation, which is the basis for effective and time-extended interaction between a mobile robot and a human. The robot uses observed information from a laser range finder to detect persons and their position relative to the robot. This information together with the motion of the robot itself is fed through a Kalman filter, which utilizes a model of the human kinematic movement to produce an estimate of the person's pose. The resulting pose estimates are used to identify humans who wish to be approached and interacted with. The behaviour of the robot is based on adaptive potential functions adjusted accordingly such that the persons social spaces are respected. The method is tested in experiments that demonstrate the potential of the combined pose estimation and adaptive behaviour approach.

## 1 Introduction

Mobile robots are moving from factory floors out into less controlled human environments such as private homes or institutions. The success of this shift relies on the robots ability to be responsive to and interact with people in a natural and intuitive manner and accordingly human-robot interaction is a novel and growing research field [1, 2].

To allow close, more effective and time-extended relationships it is first necessary to determine the persons willingness to engage in interaction, followed by a coordination in time and space that respects the persons interest and privacy. Several authors [3, 4, 5] have investigated the willingness of people to engage in interaction with robots that exhibit different expressions or follow different spatial behaviour schemes. In [6] models are reviewed that describe social engagement based the spatial relationships between a robot and a person with emphasis on the movement of the actors. Human-aware detection, tracking and navigation were discussed in [7, 8].

As a step in the direction of human aware robot behaviour, we present a novel method for inferring a human's pose from 2D laser range measurements. Here we define pose as the position, and orientation of the body. Compared to vision based pose estimation, such as [9], or 3D range scans [10], using 2D laser range scanners provide extra long range and lower computational complexity. The extra range enables the robot to detect the movement of persons moving at a higher speed. The approach takes advantage of the inherent mobility and typical sensors of a mobile robotic platform and does not require any determination of the persons facial expressions or other gestures, and hence the person does not have to be facing the robot. The method relies on an algorithm for detecting legs of persons. The algorithm has been tested in a public transit space [11].

While the focus of this paper is on the pose estimation, the use of the estimates in determining persons willingness to engage in interaction is analyzed from the recorded kinematic state of the person. By looking at knowledge from previous encounters, the robot behaviour is adjusted as described in [12]. When the persons willingness has been determined it is used as a basis for human-aware navigation using adaptive potential functions centered around the person, inspired by [7]. The navigation is adapted such that it respects the persons social spaces as discussed in [5].

The pose estimation is validated through a number of experiments. In addition, experiments indicate the effectiveness of the combined algorithm for human-aware navigation.

## 2 Materials and Methods

The basis for any interaction is the ability of the robot to detect if a person is present, and if this is the case to estimate the kinematic state of that person. The algorithm for detecting legs of persons and converting these to person position estimates is described in [13], and [14] been implemented and adapted to keep track of individual persons. The implementation has been proved robust for person speeds up to $2\frac{m}{s}$ in a real world public space setting described in [11]. The position estimates from this algorithm form the basis for deriving a pose estimate of a person.

### 2.1 Person Pose Estimation

The setup, where a person is moving around, while the robot is following, can be seen in Fig. A.1. The basic idea for estimating the pose of a person ($\theta$), is to take the position estimates from the laser range finder algorithm and combine them with robot odometry information to obtain a pose estimate. Because we have both position and velocity measurements to estimate a pose, it can not be calculated directly, and a Kalman filter is therefore used to filter the measurements. The filter produces a velocity estimate of the person relative to the robot. After this a post filter is added to obtain the person pose from the velocity estimates.

A standard discrete state space model formulation for the system is used:

$$\boldsymbol{x}(k+1) = \Phi\boldsymbol{x}(k) + \Gamma\boldsymbol{u}(k) \tag{A.1}$$

$$\boldsymbol{y}(k) = H\boldsymbol{x}(k) \quad , \tag{A.2}$$

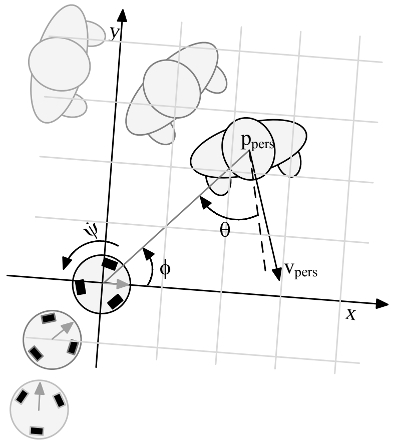


Figure A.1: Person position and pose. The state variables $\boldsymbol{p}_{pers}$ and $\boldsymbol{v}_{pers}$ hold the position and velocity of the person in the robot frame. $\theta$ is the pose of the person. The variable $\theta$ is approximately the angle between $\phi$ (the angle of the distance vector from the robot to the person) and $\boldsymbol{v}_{pers}$ (the angle of the person's velocity vector), but not entirely the same because the body is not necessarily oriented in the moving direction. $\dot{\psi}$ is the rotational velocity of the robot.

where the state is comprised of the person position and velocity and the robot velocity

$$\boldsymbol{x} = \begin{bmatrix} p_{x,pers} \\ p_{y,pers} \\ v_{x,pers} \\ v_{y,pers} \\ v_{x,rob} \\ v_{y,rob} \end{bmatrix} . \tag{A.3}$$

Here $p$ is positions and $v$ is velocities, all given in the robot coordinate frame. The position of the person relative to the robot depends both on the person velocity and the robot velocity. In this stage we omit the rotation of the robot:

$$p_{x,pers}(k+1) = p_{x,pers}(k) + T(v_{x,pers}(k) - v_{x,rob}(k)) \tag{A.4}$$

$$p_{y,pers}(k+1) = p_{y,pers}(k) + T(v_{y,pers}(k) - v_{y,rob}(k)) , \tag{A.5}$$

where $T$ is the sampling time. This yields the following state transition matrix:[1]

[1]There were too many T's in the original paper. The T's at positions (3,5) and (4,6) have been removed to make it correct here.

$$\Phi = \begin{bmatrix} 1 & 0 & T & 0 & -T & 0 \\ 0 & 1 & 0 & T & 0 & -T \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad . \tag{A.6}$$

The measurements used are the person position, and the odometry velocity data from the robot.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad . \tag{A.7}$$

To overcome the nonlinear effect the robot rotation has on the state, a measurement driven Kalman filter, which is used in [15, 16], is applied. The idea is to use sensor readings to drive the process model as an input - in this case odometry data from the robot. Using polar coordinates, the position vector of the person relative to the robot can be written:

$$\boldsymbol{p}_{pers} = \begin{bmatrix} d\cos(\phi(t)) \\ d\sin(\phi(t)) \end{bmatrix} \tag{A.8}$$

where $d$ is the distance to the person, and $\phi(t)$ is the angle to the person. The change, i.e. the derivative becomes

$$\dot{\boldsymbol{p}}_{pers} = \begin{bmatrix} \text{-}d\sin(\phi(t)) \\ d\cos(\phi(t)) \end{bmatrix} \dot{\phi}(t) \quad , \tag{A.9}$$

where $\dot{\phi}(t)$ has the opposite sign of the rotation of the robot itself $\dot{\psi}$, so the derivative becomes

$$\dot{\boldsymbol{p}}_{pers} = \begin{bmatrix} \text{-}d\sin(\phi(t)) \\ d\cos(\phi(t)) \end{bmatrix} (\text{-}\dot{\psi}) = \begin{bmatrix} p_{y,pers} \\ \text{-}p_{x,pers} \end{bmatrix} \dot{\psi} \quad . \tag{A.10}$$

Using an Euler integration, we can substitute $\Gamma\boldsymbol{u}$ in the model by

$$\Gamma\boldsymbol{u} = \begin{bmatrix} p_{y,pers}(k) \\ \text{-}p_{x,pers}(k) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} T\hat{\dot{\psi}} \quad , \tag{A.11}$$

where $\hat{\dot{\psi}}$ is the estimated robot rotation from the odometry data.

The velocity vector is not necessarily equal to the pose of the person. Consider a situation where the person is standing almost still in front of the robot, but is moving slightly backwards. This means that the velocity vector suddenly is in the opposite direction, but the actual pose is the same. Therefore the velocity estimate is filtered through a first order autoregressive filter. The filter is made with adaptive coefficients relative to the velocity. So when the person is moving fast, we rely very much on the direction of the velocity, but if the person is moving slow, we do not change the pose estimate very much and rely on the previous estimate. The autoregressive filter is implemented as

$$\theta(k+1) = \beta\theta(k) + (1-\beta)\arctan\left(\frac{v_{y,pers}}{v_{x,pers}}\right) \quad , \tag{A.12}$$

where $\beta$ is chosen relative to the absolute velocity $v$ as:

$$\beta = \begin{cases} 0.9 & \text{if } v < 0.1m/s \\ 1.04 - 1.4v & \text{if } 0.1m/s \leq v \leq 0.6m/s \\ 0.2 & \text{else} \end{cases} \quad . \tag{A.13}$$

## 2.2 Evaluating a Person's Willingness to Interact

Although the robot is not perceived as a human being when encountering people, the hypothesis is that human behavioural reactions are the same as in human to human encounters. If a person is interested, he or she will undoubtedly approach the robot in a straightforward manner. On the contrary, if in no interest the person will carefully avoid the path of the robot.

However, there may be many trajectories where the interest of the person will be difficult to determine, i.e. many valid trajectories are possible and trajectories variate for each robot-person encounter. In previous work [12] an adaptive person evaluator based on a Case Based Reasoning (CBR) system has been used. The CBR system is basically a database system which holds a number cases describing each encounter. The specification of a case is a question of determining a representative set of features connected to the event of a human robot encounter which can serve as input. In this case we use the features velocity, position and pose illustrated in Fig. A.1. The output is the Person Interest indicator, *PI*$\in [0, 1]$ which store the probability or indication of the detected person's interest to interact. The value 0 indicates no interest whereas 1 indicate interest.

The starting point of the CBR system is an empty database holding no a priori correspondence between trajectories and interest. When interacting, interest is confirmed by handing over an object to the robot. Lack of interest is triggered if no interaction has occurred after a fixed period of time. By adding cases, the system gradually learns how to decode trajectories into a interest level.

## 2.3 Human-aware Navigation

The robot behaviour is inspired by the spatial relation between humans (proxemics) as described in [17]. Hall divides the zone around a person into to four categories, 1) the public zone $> 3.6m$, 2) the social zone $> 1.2m$, the personal zone $> 0.45m$, and the intimate zone $< 0.45m$. Social spaces between robots and humans were studied in [18] supporting the use of Hall's proxemics distances and the human robot interaction is therefore designed to be able to experiment with different distances.

For modeling the robots navigation system a person centered potential field is introduced. The potential field has high values where the robot is not allowed to go, and low values where the robot should be. All navigation is done relative to the person, and hence no global positioning is needed in the proposed model. The method is described in [12], but is slightly changed in this implementation. The potential field is designed by the weighted sum of four Gaussian distributions of which one is negated. The covariances of the distributions are used to adapt the potential field according to *PI*.

The four Gaussian distributions are illustrated in Fig. A.2 and has the following functions:

**Attractor**, this is a negated distribution used to attract the robot to the person. Its variances $\sigma_x^2$ and $\sigma_y^2$ are both set to 7.5 and its covariance $\sigma_{xy}$ is set to 0.

**Rear**, this distribution ensures that the robot does not approach a person from behind. Its variances $\sigma_x^2$ and $\sigma_y^2$ are respectively set to 2 and 1 and its covariance $\sigma_{xy}$ to 0. It is only applied when the robot is behind the person.

**Parallel**, this distribution is initially placed with its major axis parallel to the $x_p$-axis in the persons coordinate frame. Its variances and covariance are adapted according to the person interested in interaction.

**Perpendicular**, this distribution is initially placed with its major axis perpendicular to the parallel distribution. Its variances and covariance are adapted according to the person interested in interaction. This distribution as well as the parallel is only applied when the robot is in front of the person.

The attractor and rear distribution are both kept constant for all instances of the person interest indication *PI*. But the parallel and perpendicular distributions are interactively scaled and rotated according to the person's changing *PI* during interaction. This means that the robot continuously adapt its behaviour to the current *PI* value. The potential functions are scaled and adjusted according to Hall's proximity distances, and the preferred robot to person encounter reported in [5].

The resulting potential field contour can be seen in Fig. A.3 for three specific values of *PI*. In the extreme case with *PI* $= 0$ the potential field will like look Fig. A.3a where

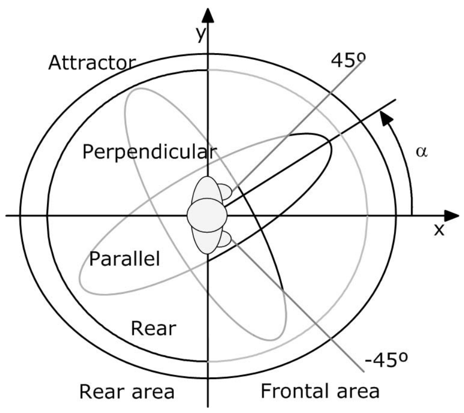


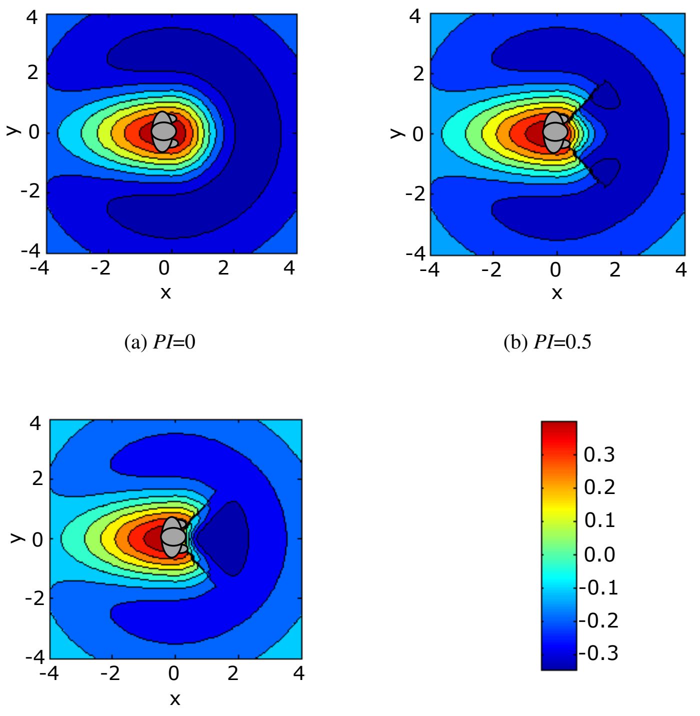
Figure A.2: Illustration of the four Gaussian distributions used for the potential function around the person. The rear area to the left of the $y$ axis. The frontal area (to the right of $y$ axis) which is divided in two, one in the interval from $[-45^\circ : 45^\circ]$ and the other in the area outside this interval. The parallel and perpendicular distributions are rotated by the angle $\alpha$.

the robot will move to the dark blue area, i.e. the lowest potential app. 2 meters in front of the person. The other end of the scale for $PI = 1$ is illustrated in Fig. A.3c, where the person is interested in interaction and as result the potential function is adapted so the robot is allowed to enter the space right in front of him or her. In between, Fig. A.3b, is the default configuration of $PI = 0.5$, in which the robot is forced to encounter the person in approximately $45^\circ$, while keeping just outside the personal zone.

Instead of just moving towards the lowest point at a fixed speed, then the gradient of the potential field is derived. This allows the robot to move fast when the potential field is steep, for example if the robot has to move fast away from a person if getting in the way. On the other hand, the robot has slow comfortable movements when it is close to where it is supposed to be, i.e. near a minimum of the field.

## 3 Experimental Setup

In order to validate all parts of the system, the above algorithms have been tested in three steps by integrating one feature at the time. First the basic person detection and pose estimation algorithm is tested. Then this is combined with the human aware navigation. Finally, also the person interest estimation has been included. Each step has been validated in real world experiment. All experiments have been done with only one human in the area, since the purpose is to demonstrate the proof of concept of the methods. However, the methods should be valid with more persons around the robot.



(a) *PI*=0 (b) *PI*=0.5

(c) *PI*=1 (d) Scale

Figure A.3: Shape of the potential function for (a) a person not interested in interaction, (b) a person considered for interaction, and (c) a person interested in interaction. The scale for the potential function is plotted to the left and the value of the person interested indicator *PI* is denoted under each plot.

## 3.1 Test Equipment and Implementation

The basis for evaluation of the proposed methods was a FESTO Robotino platform on which a head, capable of showing simple facial expressions, is mounted (see Fig. A.4). On the platform, the robot control software framework Player/Stage [19] has been implemented, which also enables simulation before real world tests. The robot is equipped with a URG-04LX line scan laser range finder and a contact to press if you are interested in interaction. The case database has been implemented using MySQL.

A 3D motion tracking system from Vicon (typically used for indoor UAV applications) has been used to validate the functionality of the algorithm through laboratory experiments.

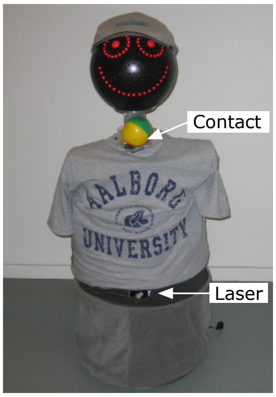


Figure A.4: The FESTO Robotino robot used for the experiments.

## 3.2 Pose Estimation

The pose estimation algorithm was validated through laboratory experiments. The pose algorithm was tested isolated from the system, i.e. all navigation and learning algorithms were disabled. During the experiment, the robot was placed on a fixed position in the lab, while a test person entered the robot field of view and wandered around following different patterns at velocities around $0.5 - 1.5\frac{m}{s}$. While the robot estimated the pose of the test person, the stationary tracking system was concurrently reading the movements.

## 3.3 Human Aware Navigation

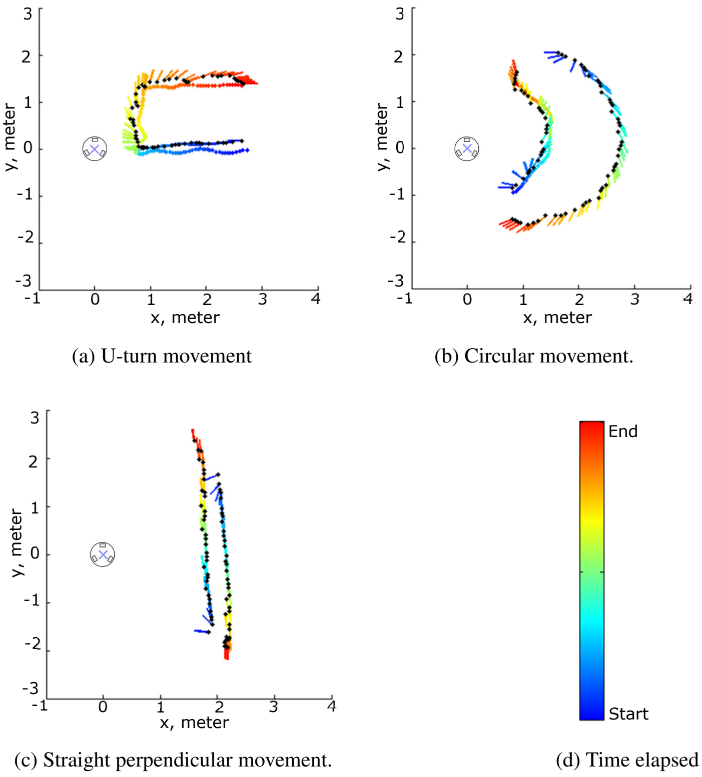
In this step, the human aware navigation algorithm was added to the pose estimation system and tested in the robot laboratory. In order to isolate the navigation algorithm, the level of *PI* was set to a fixed value through each experiment and was completed for *PI* = $\{0, 0.5, 1\}$. As in the former experiment, the stationary tracking system was set to read the position of the test person while he would approach the robot following different patterns. Since the navigation algorithm was enabled in this experiment, the movement of the robot was also tracked.

## 3.4 Integration Test

In this step, the complete system was tested, i.e. the CBR system was added. The test took place in a foyer at the University campus with an open area of 7 times 10 meters. This allowed for easily repeated tests with no interference from other objects than the test person. The contact on the robot was used to get feedback from the test persons whether they were interested in interaction or not. The test persons were asked to approach or pass by the robot in different ways. In approximately half the cases, the test person would end the trajectory by pressing the switch indicating interest in interaction. The test started with an empty database, so that the robot had no experience to start from. During the experiments the *PI* values, the position, pose estimates were logged in the database.

## 4 Results and Discussion

### 4.1 Pose Estimation



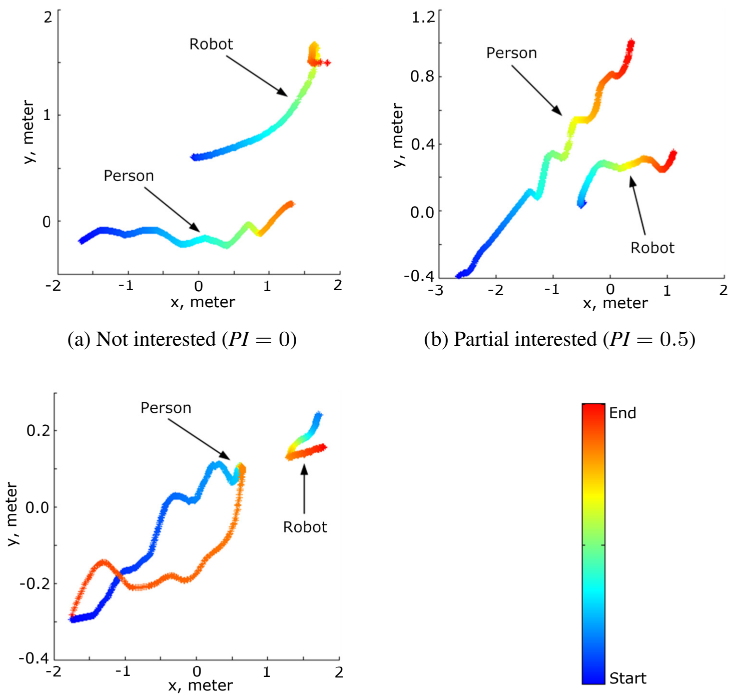
(a) U-turn movement (b) Circular movement.

(c) Straight perpendicular movement. (d) Time elapsed

Figure A.5: While the robot was standing still, three types of movement was performed. The black dots indicate the person positions, which the robot has estimated. The correct trajectory measured by the Vicon tracking system can be seen as the underlying dots in Fig. (a) and the left trajectory in Fig. (b). In the other trajectories the correct has not been plotted to avoid cluttering the graph. The time evolution is shown as colours.[a]

[a]In the original paper, some unfortunate editing had deleted some of the caption. This has been changed here.

Fig. A.5 shows the different trajectories performed. The black dots are the position estimates from the robot, and the lines extending from the dots are the corresponding pose



(a) Not interested ($PI = 0$)

(b) Partial interested ($PI = 0.5$)

(c) Interested $PI = 1$

(d) Time

Figure A.6: This figure shows the motion of the robot when the person interest indication ($PI$) is fixed at three different levels. The colour bar shows the time evolution. It can be seen that the robot keeps at a comfortable distance when the $PI$ is low, and approaches the person from the front when $PI = 1$.

estimates. Some underlying coloured dots can be seen in Fig. A.5a and in the left line in Fig. A.5b. These are the correct position estimates as measured by the motion tracking system. The correct underlying positions are omitted in the other plots, since they clutter the image unnecessarily. In Fig. A.5a, a u-turn movement is performed where the person stands still for a few seconds close to the robot in the lower left corner. This demonstrates that even though the person stands still (and might even move slightly backwards), the pose estimate keeps being correct towards the robot. When the robot recognizes a person, it assumes that the pose is close to 0, which explains the blue lines always pointing towards the robot in the beginning of a trajectory. However, as soon as the person start to walk, the pose estimate turns and follows the motion. Note that it is not the Kalman filter which estimates the velocity wrong, when the lines do not follow the trajectory exactly. But it is the autoregressive filter that does not allow the pose estimate to change too quickly. The figures show that the pose estimator works satisfactory and can be used to estimate pose in real world. Although this laboratory test confirm that the pose algorithm works as expected, the test is limited to only a few test runs and the algorithm is not really fine tuned yet.

### 4.2 Human Aware Navigation

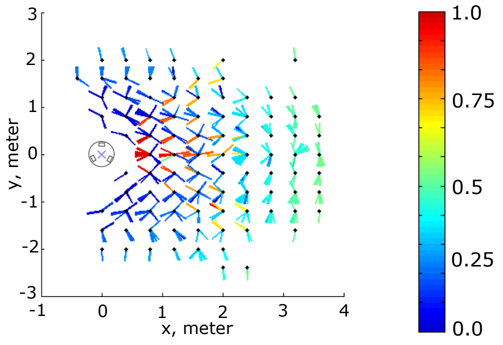
In this experiment, the robot is set to move according to prespecified person indication (*PI*) values. The person moves from the bottom left corner (in Fig. A.6a-c) towards the robot. The elapsed time is shown by the trajectory colour changing from blue to red. When $PI = 0$, it can be seen that the robot tries to get away from the person. When it reaches a comfortable distance, it settles around that position relative to the person. When the person is partially interested, the robot avoids the person and tries to stay at an angle of approximately $45°$ degrees. Finally, when the person is interested, the robot approaches the person from the front until the border of the intimate zone is reached at around $45cm$. The major colour change at the person trajectory indicates that the person has been standing still. As soon as the person starts to move away, the robot finds out that it is behind and too close to the person, so it starts to move away.

The shaky sinusoidal movement of the person trajectory is due to the tracking of the person. It is caused by the fact that the central part of the body moves like this when a human walks. The experiment proves that the potential field enables the robot to keep at the correct position relative to the person, and that the pose estimator also works when the robot is moving.

### 4.3 Integration Test

The output of the integration test was a trained CBR database. The database can be seen in Fig. A.7. The dots show recorded test person positions, which was rounded to a grid size of $40 \times 40cm$. Note that the positions are not global coordinates, but relative to the robot while it is moving. The extending lines, show the direction of the pose estimates, and the colour the corresponding *PI* value, where red indicates an interested person, and blue indicates a person which is not interested. The database shows that that persons right in front of the robot with a pose close to $0°$ is typically interested in interaction, whereas a pose pointing away from the robot indicates that the person is not interested. The results reflect the fact that the potential field makes the robot move so only persons who are estimated as interested are allowed to come close to the robot.

Clearly, the experimental work is still in its initial stage and is not exhaustive. However, the tests demonstrates the potential of the methods and of combining the pose estimation algorithm with the proposed method for human aware navigation.



(a) The case database. (b) *PI*

Figure A.7: The figure shows the values stored in the CBR system after completion of 20 test runs. Each dot represents a position of the test person in the robot coordinate frame. The direction of the pose estimate of the test person is shown by the extending line, while the level of interest (*PI*) is indicated by the colour range of the line.

## 5 Conclusions

This paper describes a new method for estimating the pose of a person in an interaction scenario with a mobile robot. The algorithm only relies on laser range finder data, which makes it applicable for moving persons at larger distances than normal vision techniques allow. A Kalman filter is used to filter the measured positions of persons within view and outputs a pose estimate.

The position and pose estimates are used in a Case Based Reasoning system to estimate the person's interest in interaction, and the spatial behaviour strategies of the robot are adapted accordingly using adaptive potential functions. The human robot interaction methodology described in this paper is supported by laboratory and real world tests which demonstrate the applicability of the pose estimator and the spatial behaviour of the robot.

The real world tests demonstrate the potential of the integrated system, which can be used for robots moving in human environments. Generally, the conducted experiments on the robots cognitive functionality show that the method of CBR implemented can advantageously be applied to a robot, which needs to evaluate the behaviour of a person.

An interesting aspect for future work would be to combine this pose estimation technique with a vision based technique. This could give a more accurate pose estimate for close interaction.

## References

[1] K. Dautenhahn, "Methodology & themes of human-robot interaction: A growing research field," *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, pp. 103–108, 2007.

[2] T. Kanda, "Field trial approach for communication robots," in *Proc. 16th IEEE International Symposium on Robot and Human interactive Communication RO-MAN 2007*, 2007, pp. 665–666.

[3] A. Bruce, I. Nourbakhsh, and R. Simmons, "The role of expressiveness and attention in human-robot interaction," in *In Proceedings, AAAI Fall Symposium*, 2001. [Online]. Available: citeseer.ist.psu.edu/article/bruce02role.html

[4] H. I. Christensen and E. Pacchierotti, "Embodied social interaction for robots," in *AISB-05*, K. Dautenhahn, Ed., Hertfordshire, April 2005, pp. 40–45.

[5] K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, E. A. Sisbot, R. Alami, and T. Siméon, "How may i serve you? a robot companion approaching a seated person in a helping context," in *HRI Human Robot Interaction '06 (HRI06)*, Salt Lake City, Utah, USA, 2006.

[6] M. Michalowski, S. Sabanovic, and R. Simmons, "A spatial model of engagement for a social robot," in *The 9th International Workshop on Advanced Motion Control, AMC06*, Istanbul, March 2006.

[7] E. A. Sisbot, A. Clodic, L. Urias, M. Fontmarty, L. Brèthes, and R. Alami, "Implementing a human-aware robot system," in *IEEE International Symposium on Robot and Human Interactive Communication 2006 (RO-MAN 06)*, Hatfield, United Kingdom, 2006.

[8] O. C. Jenkins, G. G. Serrano, and M. M. Loper, *Recognizing Human Pose and Actions for Interactive Robots*. I-Tech Education and Publishing, 2007, ch. 6, pp. 119–138.

[9] F. Dornaika and B. Raducanu, "Detecting and tracking of 3d face pose for human-robot interaction," in *Proceedings IEEE International Conference on Robotics and Automation*, 2008, pp. 1716–1721.

[10] J. Rodgers, D. Anguelov, H.-C. Pang, and D. Koller, "Object pose detection in range scan data," *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2445–2452, 2006.

[11] M. Svenstrup, T. Bak, O. Maler, H. J. Andersen, and O. B. Jensen, "Pilot study of person robot interaction in a public transit space," in *Proceedings of the International Conference on Research and Education in Robotics - EUROBOT 2008*. Heidelberg, Germany: Springer-Verlag GmbH, 2008, pp. 120–131.

[12] H. J. Andersen, T. Bak, and M. Svenstrup, "Adaptive robot to person encounter : by motion patterns," in *Proceedings of the International Conference on Research and Education in Robotics*. Springer-Verlag GmbH, 2008, pp. 13–23.

[13] D. Feil-Seifer and M. Mataric, "A multi-modal approach to selective interaction in assistive domains," in *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, 13-15 Aug. 2005, pp. 416–421.

[14] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, "Fast line, arc/circle and leg detection from laser scan data in a player driver," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 18-22 April 2005, pp. 3930–3935.

[15] M. Jun, S. Roumeliotis, and G. Sukhatme, "State estimation of an autonomous helicopter using kalman filtering," in *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 3, 17-21 Oct. 1999, pp. 1346–1353vol.3.

[16] R. van der Merwe, E. Wan, S. Julier, A. Bogdanov, G. Harvey, and J. Hunt, "Sigma-point kalman filters for nonlinear estimation and sensor fusion: Applications to integrated navigation," in *AIAA Guidance Navigation & Control Conference*, 2004.

[17] E. T. Hall, "A system for the notation of proxemic behavior," *American anthropologist*, vol. 65, no. 5, pp. 1003–1026, 1963.

[18] M. L. Walters, K. Dautenhahn, K. L. Koay, C. Kaouri, R. te Boekhorst, C. Nehaniv, I. Werry, and D. Lee, "Close encounters: Spatial distances between people and a robot of mechanistic appearance," in *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, Tsukuba, Japan, December 2005, pp. 450–455.

[19] T. Collett, B. A. MacDonald, and B. P. Gerkey, "Player 2.0: Toward a practical robot programming framework," in *In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, C. Sammut, Ed., Sydney, Australia, December 2005, http://playerstage.sourceforge.net.

# Paper B

**Adaptive Human-Aware Robot Navigation in Close Proximity to Humans**

Mikael Svenstrup, Søren Tranberg Hansen, Hans Jørgen Andersen, Thomas Bak

This paper is submitted to:
International Journal of Advanced Robotic Systems

**Abstract**

For robots, to be able coexist with people in future everyday human environments, they must be able to act in a safe, natural and comfortable way. This work addresses the motion of a mobile robot in an environment, where humans potentially want to interact with it. The designed system consists of three main components: a Kalman filter-based algorithm that derives a person's state information (position, velocity and orientation) relative to the robot; another algorithm that uses a Case-Based Reasoning approach to estimate if a person wants to interact with the robot; and, finally, a navigation system that uses a potential field to derive motion that respects the person's social zones and perceived interest in interaction.

The operation of the system is evaluated in a controlled scenario in an open hall environment. It is demonstrated that the robot is able to learn to estimate if a person wishes to interact, and that the system is capable of adapting to changing behaviours of the humans in the environment.

## 1 Introduction

Mobile robots are moving out into open-ended human environments such as private homes or even public spaces. The success of this shift relies on the robots' abilities to be responsive to, and interact with, people in a sociable, natural and intuitive manner. Research addressing these issues is labelled Human-Robot Interaction (HRI) [1, 2, 3]. The focus is often on close interaction tasks like gestures, object recognition, manipulation, face expressions or speech [4, 5, 6, 7, 8]. In shared dynamic environments, there is also a need for motion planning and navigation, which is another widely studied research topic. However, rather than just being a question of getting from A to B while avoiding obstacles, navigation algorithms have to adapt to people in the environment. As in [4], HRI and navigation must be addressed simultaneously, to support the idea of shared environments.

An example from everyday human interaction could be a person working with customer service in a mall. This person will detect other people in the environment, actively aiming to provide help. The service person will try to determine their interest or need for help and adjust actions accordingly. The service person will, over time, learn from experience and the degree of success, thereby determining subsequent behaviour in similar situations. The service person may start approaching, while determining if the person being observed is interested in help. If the person is perceived as being interested in help, the service person will typically aim to reach a position in front of the person to initiate a closer interaction. If not, the service person will get out of the way and initiate a new interaction process by looking for other people.

This may be a simplified process, but enabling a robot to support such an interaction process is not a simple task. Detection requires that the robot is able to locate people in the environment and discriminate them from obstacles. Observation requires that people can be tracked and requires an interpretation of the interest in interaction. The robot must be able to learn from previous interaction to support this interpretation. Given an interpretation of the interest to interact, the robot must adapt its navigation strategy to either approach the person in a way perceived as comfortable or get out of the way. Finally some type of closer interaction must be supported.

Detection and tracking of people, that is, estimation of the position and orientation (which combined is denoted pose) has been discussed in [9, 10]. Several sensors have been used, including 2D and 3D vision [11, 12], thermal tracking [13] or range scans [14, 15, 16]. Laser scans are typically used for person detection, whereas the combination with cameras also produces pose estimates [17, 18]. Using face detection requires the person to always face the robot, and that person to be close enough to be able to obtain a sufficiently high resolution image of the face [19], limiting the use in environments where people are moving and turning frequently. The possibility of using 2D laser range scanners provides extra long range and lower computational complexity. The extra range enables the robot to detect the motion of people further away and thus have enough time to react to people moving at a higher speed.

Interpreting another person's interest in engaging in interaction is an important component of the human cognitive system and social intelligence, but it is such a complex sensory task that even humans sometimes have difficulties with it. Research that addresses the ability to recognise human social signals and behaviours is called Social Signal Processing (SSP)[20, 21, 22]. In the example scenario, with the person from the customer service, learning was used as a way to build on previous experiences to form an interpretation of interest for a given person. Researchers have investigated Case-Based Reasoning (CBR) [23, 24]. CBR allows recalling and interpreting past experiences, as well as generating new cases to represent knowledge from new experiences. To our knowledge, CBR has not yet been used elsewhere in an HRI context but has proven successful in solving spatial-temporal problems in robotics in [25, 24, 26]. An advantage of using CBR is the simple implementation and the simple parameter tuning. Other methods, like Hidden Markov Models, have been used to learn to identify the behaviour of humans [27]. Bayesian inference algorithms and Hidden Markov Models have also successfully been applied to modelling and for predicting spatial user information [28]. To approach a person in a way that is perceived as natural and comfortable requires *human-aware navigation*. Human-aware navigation respects the person's social spaces as discussed in [29, 30, 31]. Several authors have investigated the interest of people to interact with robots that exhibit different expressions or follow different spatial behaviour schemes [32, 33, 30, 34]. In [18] models are reviewed that describe social engagement based on the spatial relationships between a robot and a person, with emphasis on the movement of the person. Although the robot is not perceived as a human when encountering people, the hypothesis is that robot behavioural reactions with respect to motion should resemble human-human scenarios. This is supported by [30, 35]. Hall has investigated the spatial relationship between humans (proxemics) as outlined in [36, 37], which can be used for Human-Robot encounters. This was also studied by Walters, et al.[35], whose research supports the use of Hall's proxemics in relation to robotics.

In this paper the focus is on detection, observation, navigation, and learning to determine the person interest in close interaction, which means that psychological studies of the naturalness and comfortable manner of the motion is out of the scope of this work. A simplified close interaction has also been implemented to complete the interaction process. We present a novel method for inferring a human's motion state (position, orientation and velocity) from 2D laser range measurements. The method relies on a fast algorithm for detecting the legs of people [17, 38], and fuses this with odometry data in a Kalman filter to obtain the motion state. Observation and learning is based on a CBR algorithm. Based on observed SSP signals, the CBR algorithm estimates a person's interest in interaction. In the given implementation, only the observed motion patterns are used as SSP information. For navigation an adaptive human-aware navigation algorithm using a potential field, based on results from [9, 39, 40], is presented. The shape of the potential field is derived from the human-human spatial relations described by Hall [37].

The algorithms have been implemented and demonstrated to work together in an open environment at the university.

## 2 Methodology

The robotic system is designed for a specific set-up, where a person encounters a robot in an open space and starts an interaction process as described above. We define the part of the interaction process starting from when the person is detected, until close interaction is initiated, as a *session*. In each session the person might, or might not, wish to engage in close interaction with the robot. The session is defined to last as long as the robot and person mutually adapts their motion according to each other, and is considered over when the person has either disappeared from the robot's field of view, or has initiated close interaction. Close interaction is, in this set-up, initiated by pressing a simple button on the robot. In this case the person is perceived by the robot to be interested in close interaction. Since SSP is difficult [21] and interest cannot be directly measured as an absolute value, we refer to interest only as it is perceived by the robot and not as the true internal state of the person. When the person has disappeared without pressing the button, they are considered to not be interested in further interaction. When a session is over, the system proceeds to detecting new people, and no further close interaction is considered.

The structure of the proposed system set-up is outlined in Fig. B.1. First, the pose of the person is estimated using a Kalman filter and an autoregressive filter. This information is sent to a person evaluation algorithm, which provides information to the navigation subsystem. The loop is closed by the robot, which executes the path in the real world and thus has an effect on how humans move in the environment.

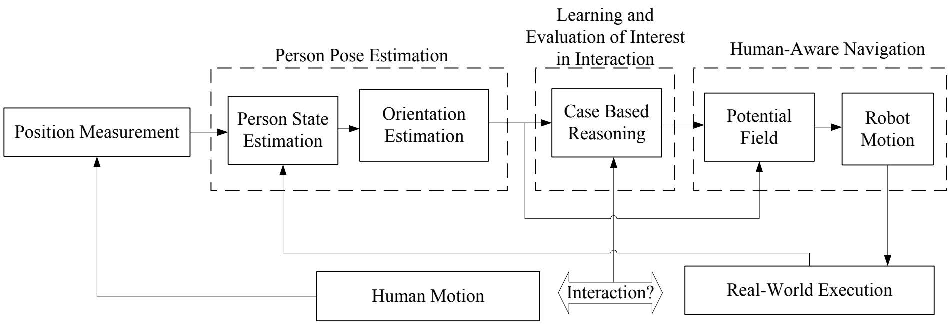


Figure B.1: An overview of how the different components are connected in the current system.

## 2.1 Person Pose Estimation

First, the pose and velocity of the people in the robot's vicinity is determined. The method is derived for only one person, but the same method can be utilised when several people are present.

We define the pose of a person as the position of the person given in the robot's coordinate system, and the angle towards the robot, as seen from the person ($p_{pers}$ and $\theta$ in Fig. B.2). The orientation $\theta$ is approximately the angle between $\phi$ (the angle of the distance vector from the robot to the person) and $\boldsymbol{v}_{pers}$ (the angle of the person's velocity vector). It is, however, only an approximation, since the orientation is not necessarily equal to the direction of the motion.

The estimation of the person pose can be broken down into three steps:

1. Measure the position of the person;
2. Use the continuous position measurements to find the state (position and velocity) of the person;
3. Use the velocity estimate to find the orientation ($\theta$) of the person.

### Position Measurements

The position measurements could, in principle, be done by any instrument, for example, external or on-board devices such as cameras, laser scanners or heat sensors. In our case we use an on-board laser range finder to find the position of the person, which has the

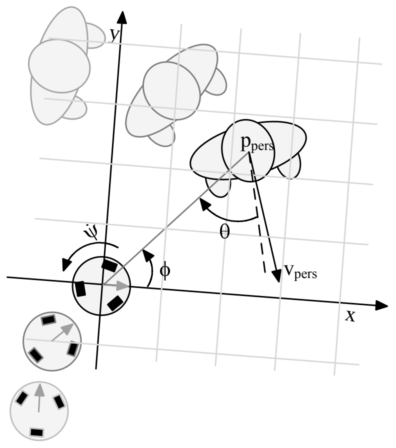


Figure B.2: The state variables $\boldsymbol{p}_{pers}$ and $\boldsymbol{v}_{pers}$ hold the position and velocity of the person in the robot's coordinate frame. $\theta$ is the orientation of the person and $\dot{\psi}$ is the rotational velocity of the robot.

advantage that it supports autonomy with no external sensors. Furthermore, processing a laser scan is computationally very efficient, compared to image processing from on-board cameras. The laser scans the area just below knee height, and a typical output from the laser range scanner is seen in Fig. B.3. An algorithm detects legs in the output data, fuses them with previous scans, pairs them and outputs estimates of where the person is. The algorithm is described in [17, 38], and the output is the position of the person in the robot's coordinate frame. Our implementation has been proved robust for tracking people at speeds of up to $2\frac{m}{s}$ in a real-world public space, which is described in [41].

### Person State Estimation

The next step is to estimate velocity vector of the person in the robot's coordinate frame. The velocity of the person cannot be calculated directly from the person position measurements since there will be an offset corresponding to the movement of the robot itself. If, for example, the person is standing still and the robot is moving, then different position measurements will be obtained, but the velocity estimate should still be zero. Therefore, the position measurements from the laser range finder algorithm and robot odometry measurements are combined to obtain a state estimate. The fusion of these measurements is done in a Kalman filter, where a standard discrete state space model formulation for the system is used:

$$\boldsymbol{x}(k+1) = \Phi\boldsymbol{x}(k) + \Gamma\boldsymbol{u}(k) \tag{B.1}$$

$$\boldsymbol{y}(k) = H\boldsymbol{x}(k) \quad , \tag{B.2}$$

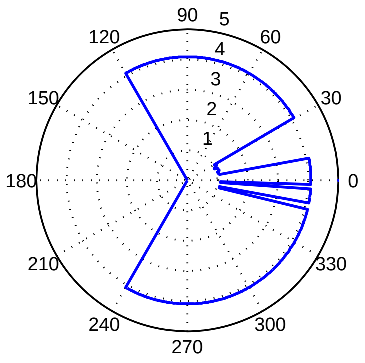


Figure B.3: An image of the ranges from one laser rangefinder scan. The distance scale on the polar plot is in metres and the angle is in degrees.

where $\boldsymbol{x}$ is the state, $\boldsymbol{y}$ is the measurements, $\boldsymbol{u}$ is the input and $\Phi, \Gamma, H$ are the system matrices explained below.

The state is comprised of the person's position, person's velocity and the robot's velocity. The measurements are the estimate of the person's position and the odometry measurements.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_{pers} \\ \boldsymbol{v}_{pers} \\ \boldsymbol{v}_{rob} \end{bmatrix} \quad , \quad \boldsymbol{y} = \begin{bmatrix} \hat{\boldsymbol{p}}_{pers} \\ \boldsymbol{v}_{odom,rob} \end{bmatrix} \quad . \tag{B.3}$$

$\boldsymbol{p}$ denotes a position vector and $\boldsymbol{v}$ denotes velocities, all given in the robot's coordinate frame. The $\hat{\boldsymbol{p}}$ is the estimate of the person's position from the person detection algorithm and $\boldsymbol{v}_{odom,rob}$ the robot's odometry measurement vector.

The position of the person in the robot's coordinate frame ($\boldsymbol{p}_{pers}$), at time $k+1$, depends both on the person's velocity, the robot's velocity and the rotation of the robot, which need to be included in the process model. However, the robot's rotation causes the vector $\boldsymbol{p}_{pers}$ to rotate, and does therefore introduce a nonlinear effect on the state. This nonlinear effect can be overcome while still maintaining the linear Kalman filter, by introducing a measurement-driven Kalman filter [42, 43]. The idea in a measurement-driven Kalman filter is to use sensor readings, in this case rotational odometry data from the robot, to drive the process model as an input. This means that the non-linearity is in the input part $\Gamma\boldsymbol{u}(k)$, and the rotation can be omitted in the state transition matrix $\Phi$. The advantage is that the input is not a part of the Kalman filter equations, and the filter remains a linear filter. Omitting rotation, $\Phi\boldsymbol{x}(k+1)$ from Eq. (B.1) can be written as:

$$\begin{bmatrix} \boldsymbol{p}_{pers}(k+1) \\ \boldsymbol{v}_{pers}(k+1) \\ \boldsymbol{v}_{rob}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{pers}(k) + T\left(\boldsymbol{v}_{pers}(k) - \boldsymbol{v}_{rob}(k)\right) \\ \boldsymbol{v}_{pers}(k) \\ \boldsymbol{v}_{rob}(k) \end{bmatrix} \quad , \tag{B.4}$$

where $T$ is the sampling time. The output matrix $H$ from Eq. (B.2) follows directly from the measurement vector in Eq. (B.3), and thus $\Phi$ and $H$ are given as:

$$\Phi = \begin{bmatrix} \boldsymbol{I} & T\boldsymbol{I} & -T\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \quad , \quad \mathbf{H} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \quad , \tag{B.5}$$

where $\boldsymbol{I}$ is the 2x2 identity matrix, and $\boldsymbol{0}$ is the 2x2 zero matrix.

The error in the linear process model, which is introduced by the rotational velocity, is compensated for by adding it to the input $\Gamma\boldsymbol{u}(k)$. Looking at Fig. B.2, it can be seen that the angle of the vector towards the person ($\boldsymbol{p}_{pers}$) is rotated by $\dot{\psi}T$ for each time step, which is also the case for the person's velocity vector ($\boldsymbol{v}_{pers}$). The effect is derived for a state vector $\boldsymbol{p}$, but applies to both $\boldsymbol{p}_{pers}$ and $\boldsymbol{v}_{pers}$. To account for the non-linearity, the input needs to compensate for the error obtained by the linear model, i.e. the difference between the rotated state vector and the vector calculated by the linear model in (B.4):

$$\Gamma\boldsymbol{u}(k) = \boldsymbol{p}_{rotated} - \boldsymbol{p}_{linear} \quad , \tag{B.6}$$

where $\boldsymbol{p}_{rotated}$ is the correct state estimate taking into account the rotation, and $\boldsymbol{p}_{linear}$ is the linear state estimate. $\boldsymbol{p}_{rotated}$ can be calculated by rotating the linear state by the angle $\dot{\psi}T$:

$$\boldsymbol{p}_{rotated} = \begin{bmatrix} \cos(\dot{\psi}T) & \sin(\dot{\psi}T) \\ -\sin(\dot{\psi}T) & \cos(\dot{\psi}T) \end{bmatrix} \boldsymbol{p}_{linear} \tag{B.7}$$

If the sample time is small compared to the rotational velocity, then a small angle approximation of the equations can be used. This means that $\cos(\dot{\psi}T) \approx 1$ and $\sin(\dot{\psi}T) \approx \dot{\psi}T$, and Eq. (B.6) can be rewritten as:

$$\Gamma\boldsymbol{u}(k) \approx \begin{bmatrix} 1 & \dot{\psi}T \\ -\dot{\psi}T & 1 \end{bmatrix} \boldsymbol{p}_{linear} - \boldsymbol{p}_{linear} = \begin{bmatrix} p_y \\ -p_x \end{bmatrix} \dot{\psi}T \quad , \tag{B.8}$$

where $p_x$ and $p_y$ are the $x$ and $y$ coordinates of the vector $\boldsymbol{p}$. Using this result to write out the full input part of the model in Eq. B.1 gives:

$$\Gamma(k)\boldsymbol{u}(k) = \begin{bmatrix} p_{y,pers}(k) \\ \text{-}p_{x,pers}(k) \\ v_{y,pers}(k) \\ \text{-}v_{x,pers}(k) \\ 0 \\ 0 \end{bmatrix} T\hat{\dot{\psi}}(k) \quad , \tag{B.9}$$

where $\hat{\dot{\psi}}$ is the measured robot rotation from the odometry data.

If $\Phi(k)$ is not constant, the Kalman gain does not converge to a steady state value. Small deviations of $\Phi(k)$ will not be a practical problem, but some infrequent long sampling intervals have been observed. In the practical implementation $\Phi(k)$ and corresponding Kalman gain is therefore calculated on-line.

### Orientation Estimation

The direction of the velocity vector found above is not necessarily equal to the orientation of the person. Consider, for example, a situation where the person is standing almost still in front of the robot, but is moving slightly backwards. This means that the velocity vector is pointing away from the robot, although the person is facing the robot. To obtain a correct orientation estimate the velocity estimate is filtered through a first order autoregressive filter, with adaptive coefficients relative to the velocity. When the person is moving quickly, the direction of the velocity vector has a large weight, but if the person is moving slowly, the old orientation estimate is given a larger weight. The autoregressive filter is implemented as:

$$\theta(k+1) = \beta\theta(k) + (1-\beta)\arctan\left(\frac{v_{y,pers}}{v_{x,pers}}\right) \quad , \tag{B.10}$$

where $\beta$ has been chosen experimentally relative to the absolute velocity $v$ as:

$$\beta = \begin{cases} 0.9 & \text{if } v < 0.1m/s \\ 1.04 - 1.4v & \text{if } 0.1m/s \leq v \leq 0.6m/s \\ 0.2 & \text{else} \end{cases} \quad . \tag{B.11}$$

In Fig. B.4 the parameter is plotted versus the velocity.

## 2.2 Learning and Evaluation of Interest in Interaction

To represent the person's interest in interaction, a continuous fuzzy variable, Person Indication (*PI*), is introduced, which serves as an indication of whether or not the person is interested in interaction. *PI* belongs to the interval $[0, 1]$, where $PI = 1$ represents the case where the robot believes that the person wishes to interact, and $PI = 0$ the case where the person is not interested.

To determine the value of *PI*, an adaptive person evaluator, based on a CBR system, is designed. The method for estimating the *PI* of a person is to observe the behaviour of the person continuously, find out if something similar has happened in previous sessions, and set *PI* according to what has happened in previous sessions. The implementation of the CBR system is basically a database system which holds a number of cases describing each session. There are two distinct stages of the CBR system operation. The first is

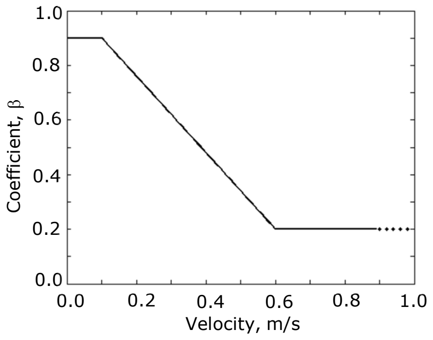


Figure B.4: The adaptive coefficient $\beta$ in the first order autoregressive filter used in the orientation estimation. The faster the person moves relative to the robot, the lower $\beta$ is and the more the filter relies on the measurements.

the evaluation of the *PI*, where the robot estimates the *PI* during a session using the experience stored in the database. The second stage is the learning stage, where the information from a new experience is used to update the database. The two stages can be seen in Fig. B.5, which shows a state diagram of the operation of the CBR system. Two different databases are used. The *Case Library* is the main database, which represents all the knowledge the robot has learned so far, and is used to evaluate *PI* during a session. All information obtained during a session is saved in the *Temporary Cases* database. After a session is over, the information from the Temporary Cases is used to update the knowledge in the Case Library.

### Database Format

Specifying a case is a question of determining a distinct and representative set of features connected to the event of a human-robot interaction session. The set of features could be anything identifying the specific situation, such as, the person's velocity and direction, position, relative position and velocity to other people, gestures, time of day, day of the week, location, height of the person, colour of their clothes, their facial expression, their apparent age and gender etc. The selected features depend on available sensors. For the experimental system considered in this paper, we consider only the motion behaviour of the person, and thus the following fields corresponding to each case will be used in the CBR database:

- CaseID, is a reference number for each case

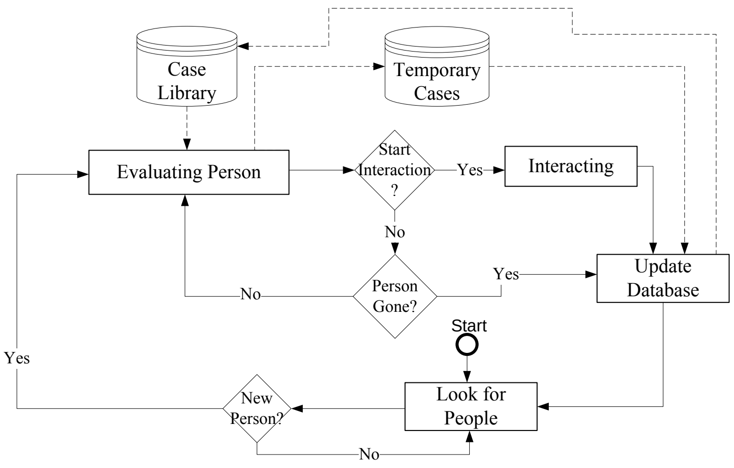


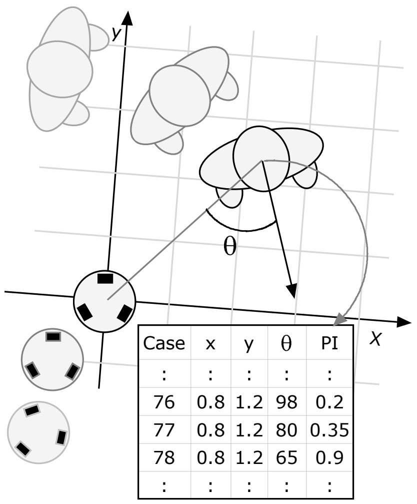
Figure B.5: Operation of the CBR system. First the robot starts to look for people. When a person is found, it evaluates whether the person wants to interact. After a potential interaction, the robot updates the knowledge library with information about the interaction session, and starts to look for other people.

- $(p_x, p_y)$, is the estimated position of the robot
- $(v_x, v_y)$, is the estimated velocity of the robot
- $\theta$, is the estimated orientation of the person
- *PI*, is the corresponding Person Indication output value.

To be able to compare cases to see if any are identical, each value is rounded to an implementation specific resolution. The database parameters are illustrated in Fig. B.6.

### Evaluation of Person Interest

Initially, when the robot locates a new person in the area, nothing is known about this person, so *PI* is assigned to the default value *PI*$= 0.5$. After this, the *PI* of a person is continuously evaluated using the Case Library. First a new case is generated by collecting the relevant set of features described in Section 2.2. The case is then compared to existing cases in the Case Library to find matching cases. When searching for matching cases in the Case Library, a similarity function must be used to define when two cases are similar enough to be interpreted as a match, since in physical system, measurements will seldom be completely identical. This means that if, for example, the distance estimates are within $10cm$, the similarity function can define the cases as matching. In our system two cases are defined as matching given that the position and orientation difference are both within



| Case | x | y | θ | PI |
|---|---|---|---|---|
| : | : | : | : | : |
| 76 | 0.8 | 1.2 | 98 | 0.2 |
| 77 | 0.8 | 1.2 | 80 | 0.35 |
| 78 | 0.8 | 1.2 | 65 | 0.9 |
| : | : | : | : | : |

Figure B.6: Input variables and the corresponding output *PI* stored in the CBR database.

a specified range limit. If a match is found in the Case Library, the *PI* is updated towards the value found in the library according to the formula

$$PI_{new} = 0.2PI_{library} + 0.8PI_{old} \quad , \tag{B.12}$$

where $PI_{new}$ is the new updated value of *PI*. This update is done to continuously adapt *PI* according to the observations, but still not trusting one single observation completely. If the robot continuously observes similar $PI_{library}$ values, the belief of the value of *PI*, will converge towards that value, e.g., if the robot is experiencing a behaviour, which earlier has resulted in interaction, lookups in the Case Library will result in $PI_{library}$ values close to 1. This means that the current *PI* will quickly approach 1 as well. After this, the case is copied to Temporary Cases, which holds information about the current session. If no match is found in the Case Library, the *PI* value is updated with $PI_{library} = 0.5$, which indicates that the robot is unsure about the intentions of the person. The case is still inserted into the Temporary Cases.

### Updating the Database

If the session resulted in close interaction, the robot assumes that the person was actually interested in the close interaction, and assumes the opposite if no close interaction occurred. This information, together with the Temporary Cases is used to revise the Case Library. The Case Library is updated according to Algorithm 2.

**Algorithm 2** Updating the Case Library from the Temporary Cases

```
UpdateDatabase()
for all TemporaryCases do
    match = FindMatchingCase()
    if match == NULL then
        match = CreateNewCase()
        SetPI(0.5)
    end if
    if Interested then
        PI(match) = PI(match) + $wL$
        if PI(match) $> 1$ then
            PI(match) = 1
        end if
    else
        PI(match) = PI(match) - $wL$
        if PI(match) $< 0$ then
            PI(match) = 0
        end if
    end if
end for
```

For each of the cases in the Temporary Cases the matching case in the Case Library is found. If there is no matching case in the Case Library, a new one is created with default *PI*$=0.5$ value. Then the associated *PI* is updated by a value of $wL$, where $w$ is a weight related to the distance to the person, and $L$ is a learning rate. Events which happen at close distances will often have more relevance and be easier to interpret than those at longer distances. Therefore, an observed case should have a larger impact the closer the person is, which is taken care of by $w$. The weight is implemented utilising the behavioural zones as designated by Hall [36] and the weight as a function of the distance is shown in (B.13) and illustrated in Fig. B.7.

$$w = \begin{cases} 1 & \text{if } d < 0.45m \\ 1.36 - 0.8d & \text{if } 0.45m \leq d \leq 1.2m \\ 0.6 - \frac{1}{6}d & \text{else} \end{cases} . \tag{B.13}$$

The learning rate $L$ controls the temporal update of *PI*. It is an adjustable parameter, which determines how fast the system is adapting to new environments. In a conservative system $L$ is low, so new observations will only affect *PI* to a limited degree. In an aggressive set-up $L$ is large and, consequently, *PI* will adapt faster. After the main Case Library has been updated, the robot is ready to start over and look for new people for potential interaction.

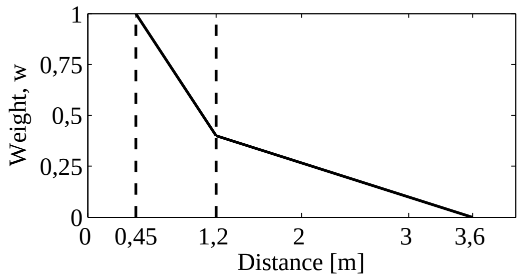


Figure B.7: The weight $w$ as a function of the distance between the robot and the test person. $w$ is a weight used for updating the level of interest *PI*.

## 2.3 Human-Aware Navigation

As written in the introduction, the human-aware navigation is inspired by the spatial relation between humans described by Hall [37]. Hall divides the area around a person into four zones according to the distance to the person:

- the public zone, where $d > 3.6m$
- the social zone, where $d > 1.2m$
- the personal zone, where $d > 0.45m$
- the intimate zone, where $d < 0.45m$

If it is most likely that the person does not wish to interact (i.e. *PI* $\approx 0$), the robot should not violate the person's personal space, but move towards the social or public zone. On the other hand, if it is most likely that the person is willing to interact or is interested in close interaction with the robot (i.e. *PI* $\approx 1$), the robot should try to enter the personal zone in front of the person. Another navigation issue is that the robot should be visible to the person, since it is uncomfortable for a person if the robot moves around, where it cannot be seen.

To allow human-aware navigation, a person-centred potential field is introduced. The potential field has high values where the robot is not allowed to go, and low values where the robot should be, or should try to move towards.

All navigation is done relative to the person(s), and hence no global positioning is needed in the proposed model. The method is inspired by [9] and further described in [44]. The potential field landscape is derived as a sum of several different potential functions with different purpose. The different functions are:

- **Attractor**. This is a negative potential used to attract the robot towards the person;

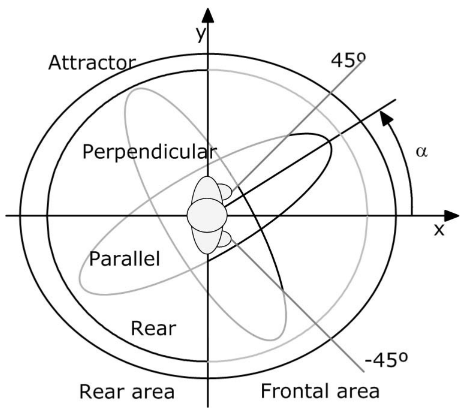


Figure B.8: The four Gaussian distributions used for the potential field around the person. The rear area is to the left of the $y$ axis. The frontal area (to the right of $y$ axis) is divided into two: one in the interval from $[-45^{\circ} : 45^{\circ}]$, and the other in the area outside this interval. The parallel and perpendicular distributions are rotated by the angle $\alpha$.

- **Rear**. This function ensures that the robot does not approach a person from behind;
- **Parallel**. This is an adaptive function, which is used to control the direction to the person and to keep a certain distance;
- **Perpendicular**. This function is also adaptive and works in cooperation with the parallel potential function.

All four potential functions are implemented as normalised bi-variate Gaussian distributions. A Gaussian distribution is chosen for several purposes. It is smooth and easy to differentiate to find the gradient, which becomes smaller and smaller (i.e. has less effect) further away from the person. Furthermore, it does not grow towards $\infty$ around 0 as, for example, a hyperbola (e.g. $\frac{1}{\boldsymbol{x}}$), which makes both computational feasible and intuitively perceivable.

The shapes of the variances of the four Gaussian distributions are illustrated in Fig. B.8, and the combined potential field is:

$$f(\boldsymbol{x}) = \sum_{k=1}^{4} c_k \exp\left(-\frac{1}{2}[\boldsymbol{x}-\mathbf{0}]^T \Sigma_k^{-1} [\boldsymbol{x}-\mathbf{0}]\right) \quad , \tag{B.14}$$

where $k = 1 \ldots 4$ is the different potential functions, $c_k$ is a normalising constant and $\boldsymbol{x}$ is the position relative to the person, where the potential function is evaluated. $\Sigma_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$ is the 2x2 covariance matrix of the distribution, which determines the shape and rotation. A way to obtain a rotated distribution while maintaining the width, is to take a standard non-rotated distribution, and rotate the covariance matrix, such that $\Sigma_k = \mathcal{R}^T \begin{bmatrix} \sigma^2_{x,std} & 0 \\ 0 & \sigma^2_{y,std} \end{bmatrix} \mathcal{R}$.

The attractor and rear distribution are both kept constant for all instances of *PI*. The parallel and perpendicular distributions are continuously adapted according to the *PI* value and Hall's proximity distances during an interaction session. Furthermore, the preferred robot-to-person encounter direction, reported in [30, 45], is taken into account by changing the width by rotation of the distributions. The width of the parallel and perpendicular distributions is adjusted by the value of the variances $\sigma^2_{x,std}$ and $\sigma^2_{y,std}$. The rotation $\alpha$ may be adapted by adjustment of the rotation matrix $\mathcal{R}$. For the parallel distribution $\sigma_{x,std} = 1$, and for the perpendicular distribution $\sigma_{y,std} = 1$. The mapping from *PI* to the respective other value of $\sigma_{x,std}$ or $\sigma_{y,std}$ is illustrated in Fig. B.9, together with the rotation angle $\alpha$. The variance cannot be zero, which is why the lowest value is $0.05$.

The adaptation of the potential field distributions enables the robot to continuously adapt its behaviour to the current interest in interaction of the person in question.

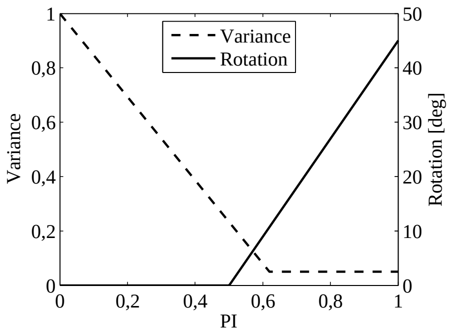
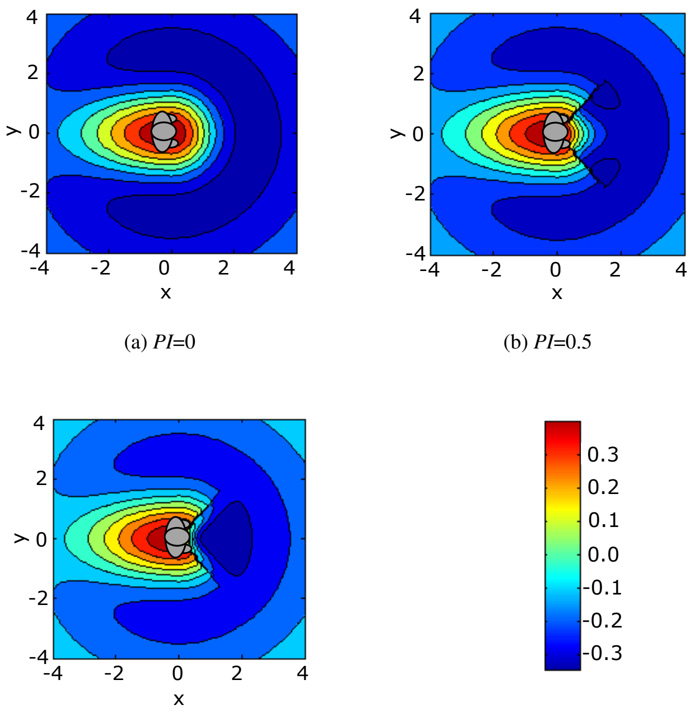


Figure B.9: Mapping of PI to the width ($\sigma^2_{std}$) along the minor axis of the parallel and perpendicular distributions, together with the rotation ($\alpha$) of the two distributions.

The resulting potential field contour can be seen in Fig. B.10 for three specific values of *PI*. With *PI* $= 0$ the potential field will look like Fig. B.10a where the robot will move to the dark blue area, i.e. the lowest potential approximately $3.6m$ in front of the person. The other end of the scale for *PI* $= 1$ is illustrated in Fig. B.10c, where the person is interested in interaction and, as a result, the potential function is adapted so the robot is allowed to enter the space right in front of the person. In between, Fig. B.10b, is the default configuration of *PI* $= 0.5$, in which the robot is forced to approach the person at approximately $45°$, while keeping just outside the personal zone.

Instead of just moving towards the lowest point at a fixed speed, the gradient of the potential field $\nabla f(\boldsymbol{x})$, is used to set a reference velocity for the robot. The gradient is



(a) *PI*=0 (b) *PI*=0.5

(c) *PI*=1 (d) Scale

Figure B.10: Shape of the potential field for (a), a person not interested in interaction, (b) a person considered for interaction, and (c) a person interested in interaction. The scale for the potential field is plotted to the right and the value of the person interested indicator *PI* is noted under each plot.

obtained from Eq. (B.14):

$$\nabla f(\boldsymbol{x}) = \sum_{k=1}^{4} -\Sigma_k^{-1}[\boldsymbol{x}-\boldsymbol{0}]c_k \exp\left(-\frac{1}{2}[\boldsymbol{x}-\boldsymbol{0}]^T\Sigma_k^{-1}[\boldsymbol{x}-\boldsymbol{0}]\right) \quad . \tag{B.15}$$

The reference velocity is then calculated as

$$\boldsymbol{v}_{ref} = k\nabla f(\boldsymbol{x}) \quad , \tag{B.16}$$

where $\boldsymbol{v}_{ref}$ is the reference velocity vector and $k$ is an adjustable parameter that controls how aggressive the motion of the robot is. This way of controlling the robot's velocity allows the robot to move quickly when the potential field is steep. On the other hand, the robot has slow comfortable movements when it is close to where it is supposed to be, i.e. near a minimum of the field.

## 3 Experimental Set-up

In [41] it is demonstrated that the person detection algorithm is working in a real-world scenario in a shopping centre. The person state estimation and human-aware navigation methods are tested in a laboratory setting in [46]. Therefore, the experiments here focus on integrating all three of the methods described above into a combined system, with the main focus on the operation of learning and evaluation of interest. The experiments are designed to illustrate the operation and the proof of concept of the combined methods. It took place in an open hall, with only one person at a time in the shared environment. This allowed for easily repeated tests with no interference from other objects than the test persons. The test persons were selected randomly from the students on campus. None had prior knowledge about the implementation of the system.

### 3.1 Test Equipment and Implementation

The robot used during the experiments was a FESTO Robotino platform, which provides omnidirectional motion. A head, which is capable of showing simple facial expressions, is mounted on the robot (see Fig. B.11). The above described algorithms have been implemented in the robot control software framework Player/Stage [47], which is implemented on the robot. The framework also enables simulation before experiments. The robot is equipped with an URG-04LX line scan laser range finder for detection and tracking of people, and a button to press for emulating the transition to close interaction (see Fig. B.11). If the test persons passed an object to the robot, they would activate the button, which was perceived as an interest in close interaction. If the test person did not pass an object (i.e. the button was not activated) within 15 seconds or disappeared from the robot's field of view, this was recognised as if no close interaction had occurred.

The CBR database has been implemented using MySQL. To make the case database tractable, the position $(p_x, p_y)$ was sampled using a grid resolution of $40cm$, and the resolution of the orientation of a person has been set to $0.2rad$. A learning rate, $L = 0.3$, which is a fairly conservative learning strategy, has been used for the experiments.

### 3.2 Test Specification

For evaluation of the proposed methods, two experiments with the combined system were performed. During both experiments the full system, as seen in Fig B.1, is used, i.e. the pose estimation and the human-aware navigation are running, as well as the interest learning and evaluation. All output values (*PI*), and input values (pose and velocity) were logged for later analysis during both experiments.

**In Experiment 1,** the objective was to see if the system was capable of learning to estimate *PI* based on interaction experience from several different people. As the number

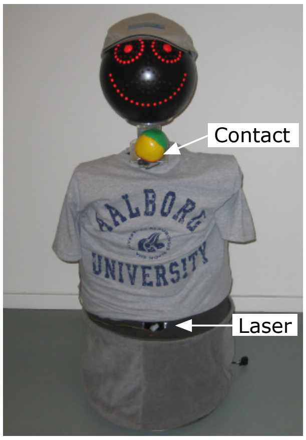


Figure B.11: The FESTO Robotino robot used for the experiments.

of cases increase, the system should be better able to estimate *PI* of a person and do it more quickly. Furthermore, the information in the CBR database should be generic, such that information obtained with some people can be used when other people occur. Starting from a CBR system with no knowledge, i.e. an empty database, a total of five test persons were asked to approach or pass the robot 12 times each using different motion patterns (see Fig. B.12). The starting and end points of each session were selected randomly, while the specific route was chosen by the test person. The random selection was designed so the test persons would end up with close interaction in $50\%$ of the sessions. In the other sessions, the test persons would pass the robot either to the left of the right without interacting.

**In Experiment 2,** the objective was to test the adaptiveness of the CBR system. The system should be able to change its estimation of *PI* over time if behaviour patterns change. A total of $36$ test approaches were performed with one test person. The test person would start randomly at P1, P2 or P3 (see Fig. B.12) and end the trajectory at P4, P5 or P6. In the first $18$ sessions the test person would indicate interest in close interaction by handing an object to the robot from P5, while in the last $18$ sessions the person did not indicate interest and the trajectory ended at P4 or P6.

## 4 Results

### 4.1 Experiment 1

As interaction sessions are registered by the robot, the database is gradually filled with cases. All entries in the database after different stages of training are illustrated by

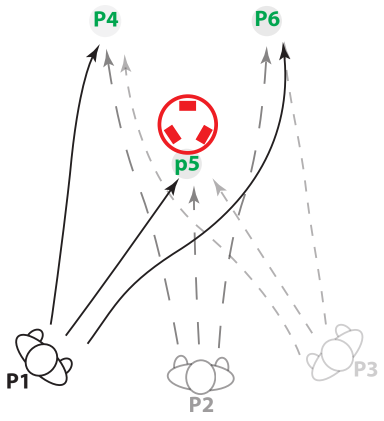
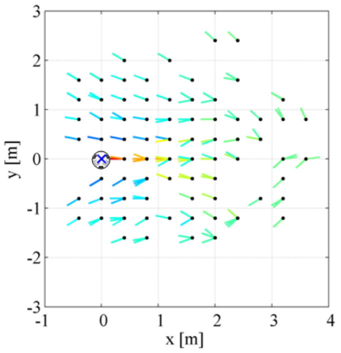


Figure B.12: Illustration of possible pathways around the robot during the experiment. A test person starts from points P1, P2 or P3 and passes through either P4, P5 or P6. If the trajectory goes through P5, a close interaction occurs by handing an object to the robot.

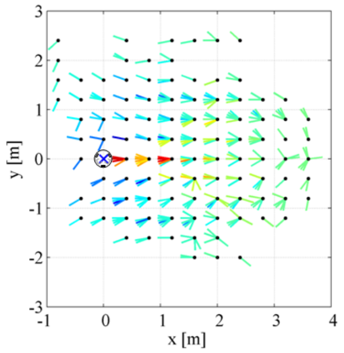
four-dimensional plots in Fig. B.13. The $(x, y)$ coordinates are the position coordinates $(p_x, p_y)$ of the person in the robot's coordinate frame, as estimated by the pose estimation algorithm. The coordinates are the dots in the $40$ x $40cm$ grid. At each position, the orientation of the person $(\theta)$ is illustrated by a vector. The colour of the vector denotes the value of *PI*. Blue indicates that the person does not wish close interaction, while red indicates that the person wishes to engage in close interaction, i.e. *PI*$= 0$ and *PI*$= 1$ correspondingly. A green vector indicates *PI*$= 0.5$.

Figs. B.13(a-c) show how information in the database evolves during the experiment. Fig. B.13a shows the state of the database after the first person has completed the 12 sessions. Here, the database is seen to be rather sparsely populated with *PI* values mostly around $0.5$, which means that the CBR system is not well trained yet. Fig. B.13b shows the state after 3 test persons have completed the 12 sessions and, finally, Fig. B.13c shows all cases (around $500$) after all 5 test persons have completed the sessions. Here the database is more densely populated, and *PI* values are in the whole range from $0$ to $1$.

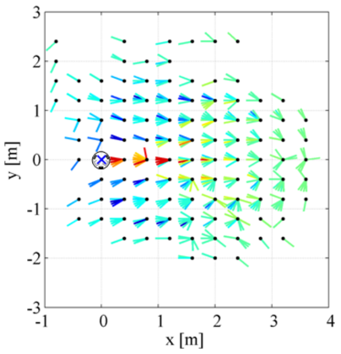
In Fig. B.14, the *PI* development for six individual sessions, are plotted as a function of time. This is done for test persons 1, 3 and 5. *PI* is plotted twice for each test person: once for a randomly selected session where the test person wishes interaction with the robot; and once for a randomly selected session where the test person passes the robot with no interest in interaction. For the first test person, *PI* increases to a maximum around 0.65 for a session ending with a close interaction. For the same test person, *PI* drops to a minimum of 0.48 for a session where no close interaction occurs. For the 3rd test person, *PI* ends with a value around 0.9 for a session where close interaction has occurred, while



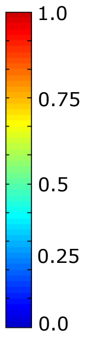
(a) After 1st test person



(b) After 3rd test person



(c) After 5th test person



(d) *PI*

Figure B.13: The figures show the values stored in the CBR system after completion of the 1st, 3rd and 5th test person. Note that the robot is located at the origin (0,0), since the measurements are in the robot's coordinate frame, which follows the robot's motion. Each dot represents a position of the test person in the robot's coordinate frame. The direction of the movement of the test person is represented by a vector, while the level (*PI*) is indicated by the colour range.

$PI = 0.35$ for a session where no close interaction has occurred. For the last test person, *PI* rapidly increases to a value around 1 for a session where close interaction occurs, and has *PI* around 0.18 when the person is not interested in interaction. Generally *PI* is estimated more quickly and with more certainty the more the system is trained.

## 4.2 Experiment 2

Experiment 2 tests the adaptiveness of the CBR system. In order to see the change in the system over time, the average *PI* value in the database after each session is calculated.

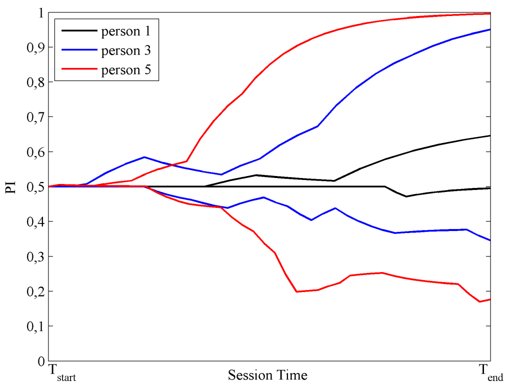


Figure B.14: *PI* as a function of the session time for three different test persons. For each test person, *PI* is plotted for a session where close interaction occurs and for a session where no close interaction occurs. The $x$-axis shows the session time. The axis is scaled such that the plots have equal length.

The averages have been calculated as an average for three different areas (see Fig. B.15), to be able to compare how the database changes in different areas relative to the robot. The areas are:

- **Area 1:** The frontal area just in front of the the robot.
- **Area 2:** A small area around the robot, which includes some of the frontal area, and some to the sides as well.
- **Area 3:** All cases stored in the database.

Fig. B.16 shows the development of the average values of *PI* for the 36 interaction sessions for one person.

As can be seen from Fig. B.16, the average value of *PI* increases for the first 18 sessions, where the person is interested in close interaction. This is especially the case for areas 1 and 2, which have a maximum value at 0.9 and 0.85 respectively, but less for area 3 (around 0.65). After 18 sessions, there is a bend on the graph and *PI* starts to drop for all areas. Most notably, area 1 drops to a minimum of 0.39 after 36 sessions.

## 5 Discussion

As can be seen in Figs. B.13(a-c), the number of plotted database entries increases as more interaction sessions occur. This shows the development of the CBR system, and clearly illustrates how the CBR system gradually learns from each person interaction session.

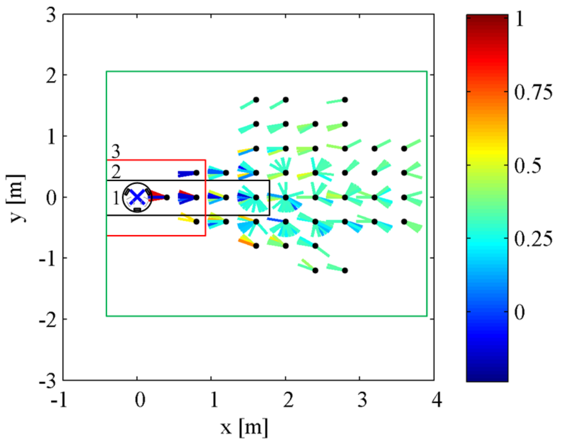


Figure B.15: A snapshot of the database after the second experiment was done. It shows how the mean value for *PI* is calculated for three areas: 1) the frontal area; 2) the small area and; 3) for all cases. The development of the mean values over time for all three areas are illustrated in Fig. B.16

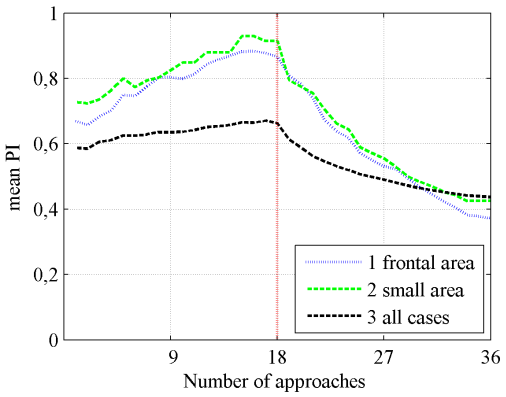


Figure B.16: Graph of how the average of *PI* evolves for the three areas indicated in Fig. B.15. 36 person interaction sessions for one test person is performed. The red vertical line illustrates where the behaviour of the person changes from seeking interaction to not seeking interaction.

The number of new cases added to the database is highest at the beginning of the training period where few (or no) case matches are found. As the training continues, the number of new cases added to the database is reduced as matching cases are found. The growth of the database, when training, depends on the resolution of the selected case features and the time and complexity of the training scenario. Based on current experiments there are no indications that the database will grow towards being inappropriately large. In Figs. B.13(a-c), the vectors are gradually turning from either red or blue to green as distance increases. This is expected, as the weight, which determines how much *PI* is changed for each update, is a function of the distance between the robot and the test person (see Fig. B.7). This is reasonable as it gets more difficult to assess human interest at long distance.

In all three figures (Figs. B.13(a-c)), the vectors in the red colour range (high *PI*) are dominant when the orientation of the person is towards the robot, while there is an excess of vectors not pointing directly towards the robot in the blue colour range (low *PI*). This reflects that a person, who wishes to interact, has a trajectory moving towards the robot, which is as expected for a normal human interaction process.

Fig. B.14 shows the development of *PI* over time for six individual sessions with positive and negative interest in interaction, respectively. It can be seen how maximum and minimum values for *PI* increase as more test persons have been evaluated. After evaluating one test person, the robot has gathered very little interaction experience, and has difficulties in determining the correspondence between motion pattern and end result - hence *PI* stays close to 0.5. After the third test person, the robot has gathered more cases and, therefore, improves in estimating the outcome of the behaviour. For the last test person, the robot is clearly capable of determining what will be the outcome of the interaction session. Each session takes between 2 and 4 seconds depending on the velocity of the user. Changes in estimation of *PI* can be seen in the first quartile of each session, while maximum (or minimum) is not reached before the fourth quartile.

Fig. B.16 shows the development of the average *PI* value over time. It can be seen that *PI* changes more for areas 1 and 2 close to the robot, than for area 3. This is because most cases will be close to 0.5 at large distances owing to the distance weight function in Eq. (B.13). Furthermore, in most sessions the person's trajectory goes through the frontal area, thereby having the highest number of updates of *PI*. Fig. B.16 illustrates that the database quickly starts to adapt to the new environment, when the test person changes the behaviour to no interaction after the first 18 sessions.

In short, the experiments show that:

- Determination of *PI* improves as the number of CBR case entries increases, which means that the system is able to learn from experience.
- The CBR system is independent of the specific person, such that experience based

on motion patterns of some people, can be used to determine the *PI* of other people.

- The algorithm is adaptive, when the general behaviour of the people changes.

### 5.1 General Discussion on Reasoning System and Adaptive Behaviour

Generally, the conducted experiments show that CBR can be applied advantageously to a robot, which needs to evaluate the behaviour of a person. The method for assessment of the person's interest in interaction with the robot is based on very limited sensor input. This is encouraging as the method may easily be extended with support from other sensors, such as computer vision, acoustics etc.

The results demonstrate how, by fairly simple training, a robot can learn to estimate the interaction interest of a person. Such training may be used to give the robot an initial database that may later be refined to the specific situation in which it operates.

The proposed method for human-aware navigation demonstrates how Hall's zones, together with more recent results about the preferred robot encounter, may be used for the design of an adaptive motion strategy.

The three methods - person pose estimation, learning and evaluation of interest in interaction, and human-aware navigation - are decoupled in the sense that they only have a simple interface with each other. This opens a way of using the methods separately with other control, navigation or interaction strategies.

This work shows that by coupling the CBR system with human-aware navigation, the result is an adaptive robot behaviour respecting the personal zones depending on the person's interest to interact - a step forward from previous studies [9, 39].

## 6 Conclusions

In this work, we have described an adaptive system for natural motion interaction between mobile robots and humans. The system forms a basis for human-aware navigation respecting a person's social spaces. The system consists of three independent components:

- A new method for pose estimation of a human, by using laser rangefinder measurements.
- Learning human behaviour using motion patterns and Case-Based Reasoning (CBR).
- A human-aware navigation algorithm based on a potential field.

Pose estimates are used in a CBR system to estimate the person's interest in interaction, and the spatial behaviour strategies of the robot are adapted accordingly using adaptive potential functions.

The evaluation of the system has been conducted through two experiments in an open environment. The first of the two experiments of the combined system shows that the CBR system gradually learns from interaction experience. The experiment also shows how motion patterns from different people can be stored and generalised in order to predict the outcome of an interaction session with a new person.

The second experiment shows how the estimated interest in interaction adapts to changes in behaviour of a test person. It is illustrated how the same motion pattern can be interpreted differently after a period of training.

The presented system is a step forward in creating socially intelligent robots, capable of navigating in an everyday environment and interacting with human beings by understanding their interest and intention. In the long-term perspective, the results could be applied to service or assistive robots in e.g. health-care systems.

In this paper, the potential field is used to steer the robot around one person, who might or might not be interested in interaction. But the potential field could also be used for navigation in populated areas with more people present. This could, for example, be in a pedestrianised street, where the robot has to move around.

## References

[1] K. Dautenhahn, "Methodology & themes of human-robot interaction: A growing research field," *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, pp. 103–108, 2007.

[2] C. Breazeal, *Designing Sociable Robots*. Cambridge, MA, USA: MIT Press, 2002.

[3] T. Kanda, "Field trial approach for communication robots," in *Proceedings of 16th IEEE International Symposium on Robot and Human interactive Communication RO-MAN 2007*, 2007, pp. 665–666.

[4] P. Althaus, H. Ishiguro, T. Kanda, T. Miyashita, and H. Christensen, "Navigation for human-robot interaction tasks," in *Proceedings of IEEE International Conference on Robotics and Automation, 2004. ICRA '04.*, vol. 2, Apr 26-May 1, 2004, pp. 1894–1900.

[5] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[6] R. Stiefelhagen, H. Ekenel, C. Fugen, P. Gieselmann, H. Holzapfel, F. Kraft, K. Nickel, M. Voit, and A. Waibel, "Enabling multimodal human-robot interaction for the karlsruhe humanoid robot," *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, vol. 23, no. 5, pp. 840–851, Oct. 2007.

[7] R. Gockley, J. Forlizzi, and R. Simmons, "Interactions with a moody robot," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. ACM, 2006, p. 193.

[8] P. Kahn, H. Ishiguro, B. Friedman, and T. Kanda, "What is a human? toward psychological benchmarks in the field of human-robot interaction," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006*, 2006, pp. 364–371.

[9] E. A. Sisbot, A. Clodic, L. Urias, M. Fontmarty, L. Brèthes, and R. Alami, "Implementing a human-aware robot system," in *IEEE International Symposium on Robot and Human Interactive Communication 2006 (RO-MAN 06)*, Hatfield, United Kingdom, 2006.

[10] O. C. Jenkins, G. G. Serrano, and M. M. Loper, *Recognizing Human Pose and Actions for Interactive Robots*. I-Tech Education and Publishing, 2007, ch. 6, pp. 119–138.

[11] F. Dornaika and B. Raducanu, "Detecting and tracking of 3d face pose for human-robot interaction," in *Proceedings IEEE International Conference on Robotics and Automation*, 2008, pp. 1716–1721.

[12] R. Munoz-Salinas, E. Aguirre, M. Garcia-Silvente, and A. Gonzalez, "People detection and tracking through stereo vision for human-robot interaction," *MICAI 2005: Advances in Artificial Intelligence*, vol. 3789/2005, pp. 337–346, 2005.

[13] G. Cielniak, A. Treptow, and T. Duckett, "Quantitative performance evaluation of a people tracking system on a mobile robot," in *Proc. 2nd European Conference on Mobile Robots*, 2005.

[14] J. Rodgers, D. Anguelov, H.-C. Pang, and D. Koller, "Object pose detection in range scan data," *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2445–2452, 2006.

[15] A. Fod, A. Howard, and M. J. Mataric, "Laser-based people tracking," in *In Proc. of the IEEE International Conference on Robotics & Automation (ICRA*, 2002, pp. 3024–3029.

[16] R. Kirby, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," in *Proceedings of Human-Robot Interaction*, March 2007, pp. 17–24.

[17] D. Feil-Seifer and M. Mataric, "A multi-modal approach to selective interaction in assistive domains," in *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, 13-15 Aug. 2005, pp. 416–421.

[18] M. Michalowski, S. Sabanovic, and R. Simmons, "A spatial model of engagement for a social robot," in *The 9th International Workshop on Advanced Motion Control, AMC06*, Istanbul, March 2006.

[19] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lomker, G. Fink, and G. Sagerer, "Person tracking with a mobile robot based on multi-modal anchoring," *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN), 2002.*, vol. 1, pp. 423–429, 2002.

[20] A. Vinciarelli, M. Pantic, and H. Bourlard, "Social signal processing: Survey of an emerging domain," *Image and Vision Computing*, vol. 27, no. 12, pp. 1743–1759, 2009.

[21] A. Vinciarelli, M. Pantic, H. Bourlard, and A. Pentland, "Social signal processing: state-of-the-art and future perspectives of an emerging domain," in *Proceedings of the 16th ACM international conference on Multimedia*. ACM, 2008, pp. 1061–1070.

[22] Z. Zeng, M. Pantic, G. Roisman, and T. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 39–58, 2008.

[23] J. Kolodner, *Case-based reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[24] A. Ram, R. C. Arkin, K. Moorman, and R. J. Clark, "Case-based reactive navigation: A case-based method for on-line selection and adaptation of reactive control parameters in autonomous robotic systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, pp. 376–394, June 1997.

[25] M. Likhachev and R. Arkin, "Spatio-temporal case-based reasoning for behavioral selection," in *Proc. IEEE International Conference on Robotics and Automation, ICRA*, vol. 2, 2001, pp. 1627–1634.

[26] I. Jurisica and J. Glasgow, "Applying case-based reasoning to control in robotics," in *3rd Robotics and Knowledge-Based Systems Workshop, St. Hubert Quebec*, 1995.

[27] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding human intentions via hidden markov models in autonomous mobile robots," in *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. New York, NY, USA: ACM, 2008, pp. 367–374.

[28] D. A. V. Govea, "Incremental learning for motion prediction of pedestrians and vehicles," Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2007.

[29] M. L. Walters, K. Dautenhahn, R. te Boekhorst, K. L. Koay, C. Kaouri, S. Woods, C. Nehaniv, D. Lee, and I. Werry, "The influence of subjects' personality traits on personal spatial zones in a human-robot interaction experiment," in *Proc. IEEE Roman*, Hashville, August 2005, pp. 347–352.

[30] K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, E. A. Sisbot, R. Alami, and T. Siméon, "How may i serve you? a robot companion approaching a seated person in a helping context," in *HRI Human Robot Interaction '06 (HRI06)*, Salt Lake City, Utah, USA, 2006.

[31] L. Takayama and C. Pantofaru, "Influences on proxemic behaviors in human-robot interaction," in *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on intelligent robots and systems*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 5495–5502.

[32] A. Bruce, I. Nourbakhsh, and R. Simmons, "The role of expressiveness and attention in human-robot interaction," in *In Proceedings, AAAI Fall Symposium*, 2001. [Online]. Available: citeseer.ist.psu.edu/article/bruce02role.html

[33] H. I. Christensen and E. Pacchierotti, "Embodied social interaction for robots," in *AISB-05*, K. Dautenhahn, Ed., Hertfordshire, April 2005, pp. 40–45.

[34] N. Hanajima, Y. Ohta, H. Hikita, and M. Yamashita, "Investigation of impressions for approach motion of a mobile robot based on psychophysiological analysis," in *IEEE International Workshop on Robots and Human Interactive Communication ROMAN 2005*, August 2005, pp. 79–84.

[35] M. L. Walters, K. Dautenhahn, K. L. Koay, C. Kaouri, R. te Boekhorst, C. Nehaniv, I. Werry, and D. Lee, "Close encounters: Spatial distances between people and a robot of mechanistic appearance," in *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*, Tsukuba, Japan, December 2005, pp. 450–455.

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

[37] E. T. Hall, "A system for the notation of proxemic behavior," *American anthropologist*, vol. 65, no. 5, pp. 1003–1026, 1963.

[38] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, "Fast line, arc/circle and leg detection from laser scan data in a player driver," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 18-22 April 2005, pp. 3930–3935.

[39] E. A. Sisbot, R. Alami, T. Siméon, K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, and C. Nehaniv, "Navigation in the presence of humans," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids 05)*, Tsukuba, Japan, December 2005.

[40] B. Fajen and W. Warren, "The behavioral dynamics of steering, obstacle avoidance, and route selection." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 29, no. 1, pp. 343–362, 2003.

[41] M. Svenstrup, T. Bak, O. Maler, H. J. Andersen, and O. B. Jensen, "Pilot study of person robot interaction in a public transit space," in *Proceedings of the International Conference on Research and Education in Robotics - EUROBOT 2008*. Heidelberg, Germany: Springer-Verlag GmbH, 2008, pp. 120–131.

[42] M. Jun, S. Roumeliotis, and G. Sukhatme, "State estimation of an autonomous helicopter using kalman filtering," in *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 3, 17-21 Oct. 1999, pp. 1346–1353vol.3.

[43] R. van der Merwe, E. Wan, S. Julier, A. Bogdanov, G. Harvey, and J. Hunt, "Sigma-point kalman filters for nonlinear estimation and sensor fusion: Applications to integrated navigation," in *AIAA Guidance Navigation & Control Conference*, 2004.

[44] H. J. Andersen, T. Bak, and M. Svenstrup, "Adaptive robot to person encounter : By motion patterns," in *Proceedings of the International Conference on Research and Education in Robotics*. Springer-Verlag GmbH, 2008, pp. 13–23.

[45] S. Woods, M. L. Walters, K. Koay, and K. Dautenhahn, "Methodological issues in hri: A comparison of live and video-basedmethods in robot to human approach direction trials," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06)*, Hatfield, UK, September 2006, pp. 51–58.

[46] M. Svenstrup, S. T. Hansen, H. J. Andersen, and T. Bak, "Pose estimation and adaptive robot behaviour for human-robot interaction," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation. ICRA 2009.*, Kobe, Japan, May 2009, pp. –.

[47] T. Collett, B. A. MacDonald, and B. P. Gerkey, "Player 2.0: Toward a practical robot programming framework," in *In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, C. Sammut, Ed., Sydney, Australia, December 2005, http://playerstage.sourceforge.net.

# Paper C

## An Adaptive Robot Game

Søren Tranberg Hansen, Mikael Svenstrup, Lars Dalgaard

This paper is published in:
Proceedings of ISR/ROBOTIK 2010

**Abstract**

The goal of this paper is to describe an adaptive robot game, which motivates elderly people to do a regular amount of physical exercise while playing. One of the advantages of robot based games is that the initiative to play can be taken autonomously by the robot. In this case, the goal is to improve the mental and physical state of the user by playing a physical game with the robot. Ideally, a robot game should be simple to learn but difficult to master, providing an appropriate degree of challenge for players with different skills. In order to achieve that, the robot should be able to adapt to the behavior of the interacting person. This paper presents a simple ball game between a single player and a mobile robot platform. The algorithm has been validated using simulation and real world experiments.

## 1 Introduction

Based on the demographic development in most western countries, it has been predicted that the number of people with mental and/or physical disabilities will increase while the amount of people to take care of them will decrease [1], [2]. Digital games hold a significant promise for enhancing the lives of seniors, potentially improving their mental and physical wellbeing, enhancing their social connectedness, and generally offering an enjoyable way of spending time [3]. It has been shown that mental and physical health can be improved through a small amount of physical exercises [4], [5], and e.g. Nintendo Wii has been suggested as a means to increase physical activity among elderly [6], [7].

In this paper we introduce a physical game which is facilitated and initiated using a mobile robot. A principal question is how to design a robot based game which ensures engagement of the participating players. Many known games are derived from a pursuit-evasion scenario e.g. the child games robbers and cops and the game of tag [8]. In this paper, we describe a simple pursuit and evasion problem played between a single player and a mobile robot. The robot will initiate the game by searching for a potential player in a room and hand over a ball. After that, the player should try to hand back the ball, while the robot should try to avoid receiving the ball.

Motivating elderly to move physically by playing a game is related to Persuasive technology which is defined as technology designed to change attitudes or behaviors of the users through persuasion and social influence, but not through coercion [9]. A similar term is Captology, which is an acronym for computers as persuasive technologies [10]. This term however, is not used as often as Persuasive Technology or Persuasive Design which is the term we will use here. Successful games are often characterized using the concept Flow, as proposed by Csíkszentmihály [11]. Flow is a mental state which can occur when there is an appropriate balance between challenge and skill. As the cognitive and physical capabilities of the users are expected to vary from each person, the robot should adapt the difficulty of the game to the end user.

The goal is to motivate elderly to do physical exercises in a fun and social manner by facilitating and initiating a simple physical game using a robot. The robot automatically initiates the game by autonomously approaching the user. This is a difference from using video games like Nintendo Wii, which facilitate games but does not itself initiate a game.

In this paper, we first outline the theory about persuasive design and the concept of flow. Next we explain the game algorithm, and demonstrate how it works when implemented in a physical robot.

## 2 Theory

The fundamental concept of Persuasive Design (PD) is persuasion, which is defined by Fogg as an attempt to change attitudes or behaviors or both (without using coercion or deception) [9]. The theoretic background is based on Computer Science and Social Psychology and has been developed mainly through empiric studies. Results from HCI, has shown that e.g. a computer can act as a social character, because it has some characteristics which make us behave as if it was a real person. We know it is piece of a technology, but can still feel happy about it or get angry with it [12]. According to Fogg, this effect is amplified if the system shows social signals we know from interaction with other people, and even more so if the system has a personality the reminds us of our own.

Figure C.1 shows how PD has been defined as a field where persuasion and computer technology overlap. The model was introduced in 1997 and has been continuously enhanced as new technologies emerge [10]. The focus of the theory is technology designed with persuasive intention.

We here extend the list of technologies in Figure C.1 to also include robots. The robot should act a social character which invites the users to play a physical game. By doing this, the user will be more mentally and physically active than would be the case without the robot. In order for people to be motivated to play the game it should appeal to the specific user. Ideally the player should be in the state of flow while playing, being a feeling characterized by great absorption and engagement as proposed by Csíkszentmihály [11]. As illustrated in Figure C.2, flow cannot occur if the task is too easy or too difficult.

In the state T1, your skills are not developed, but the challenge is not impossible. The difficulty of the challenge is in an appropriate relation to your (undeveloped) skills, and you are in a state of flow. T2 is the situation where you develop your skills to a level where the challenge becomes too easy and therefore boring. In T3, the difficulty of the challenge is higher than your skills. This leads to discontent and frustration. Common for the state T2 and T3 is that in the long run, they are unsatisfying. In order to enter the state of flow, the difficulty of the challenge has to be changed or you have to improve

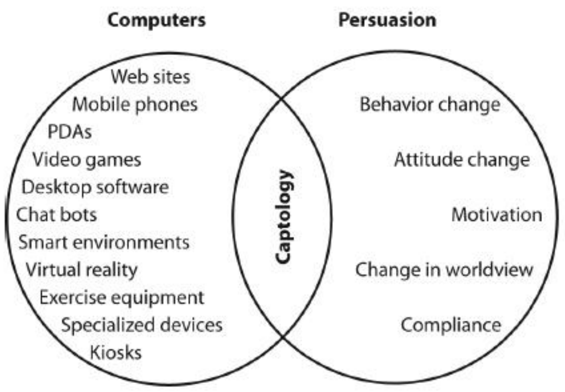


Figure C.1: Illustration of how Captology (or PD) is defined as an overlap between computers and persuasion

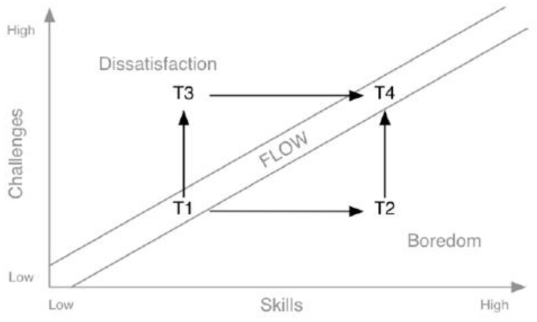


Figure C.2: Illustration of the relation between skill and challenge. In state T1 and T4, there is a balance between skill and challenge and the player is in the state of flow. In state T2 and T3 there is no balance, and the player is either bored or frustrated, correspondingly

your skills. T4 is also a flow state, but in a more complex situation than in T1. It is not a stable state because your skills will keep developing [13].

## 3 Implementation of the game

The game presented here, is based on a simplified pursuit and evasion scenario with a single pursuer (a human player) and a evader (a mobile robot). The player should try to hand over a ball to the robot, while the robot should try to avoid receiving the ball. Depending on the skill of the player, the robot should make it more or less difficult to hand the ball back. During a game, the robot can be in the following states:

- Roaming. If no player is detected, the robots should search for a player by moving randomly around until a person is spotted.
- Approach. When a player is detected, the robot invites to play a game by approaching the player from the front.
- Avoid. When the player has accepted to play a game by picking up the ball from the robot, the robot initially moves fast backwards away from the player for 3 seconds.
- Evaluating. The robot keeps avoiding the player with a distance and velocity that corresponds to the estimated skill of the player. When the ball has been handed back, the game is complete and the robot will go to the state Avoid and thereafter Roaming.

A more detailed outline of the game algorithm is sketched in Algorithm 3.

## 3.1 Player Skill Indication (*PSI*)

In order for the robot to adapt the challenge to the individual player, is should have a have an estimate of the player's skill but also information about the specific player's style of playing, i.e. the physical behavior pattern of the player.

The skill of a player is annotated using the parameter *PSI*, which means Player Skill Indication. $PSI \in [0;1]$ is a fuzzy predicate, which gives an indication of the skill of the current player. When $PSI \approx 1$, the robot believes the player is skilled, and that the player is likely to complete a game within a fixed evaluation period. When *PSI* is close or equal to $0$, the robot thinks the player is less skilled, and thereby less likely to complete the game within the time period. *PSI* is updated continuously throughout the game (line 20 in Algorithm I) , so when a specific player gets better at playing *PSI* will increase.

The rate by which *PSI* increases or decreases depends not only on the skills of the player, but is a function of the learning rate parameter, $L$. The learning rate $L$ is set high if you the game should adapt quickly to changes in the player's skill, but low otherwise.

## 3.2 Learning usning Case Based Reasoning (CBR)

The robot should learn about the specific player's style of playing with the robot, and therefore the skill is associated with the physical spatio-temporal behavior of the person. To incorporate the ability to learn the behavior pattern of the player, we have selected to use Case Based Reasoning (CBR). CBR allows recalling and interpreting past experiences, as well as generating new cases to represent knowledge from new experiences [14]. CBR has been proven successful solving spatial-temporal problems in robotics in [15] and is characterized by its adaptiveness, making it well suited for implementing an adaptive behavior on a human interactive robot. The CBR system is basically a database

**Algorithm 3**

```
Main()
 1: loop
 2:     Roam()
 3:     Approach()
 4:     if Ball just picked up then
 5:         Avoid()
 6:     end if
 7:     GameResult = Evaluate()
 8:     UpdateCbrDatabase(GameResult)
 9: end loop
Roam()
10: while Person not detected do
11:     drive randomly around
12: end while
Approach()
13: while Ball is on the robot do
14:     Approach player to invite to a game
15: end while
Avoid()
16: Avoid player for 3 seconds
Evaluate()
17: PSI = 0.5
18: while Time not expired do
19:     Move according to PSI value
20:     Update PSI using CBR database
21:     if Ball returned to robot then
22:         Avoid()
23:         return Positive
24:     end if
25: end while
26: return Negative
```

describing each encounter. Specifying a case in CBR is a question of determining a distinct and representative set of features, in our case *PSI*, position, pose and the id of the person. While playing, cases are continuously inserted, retrieved and updated. In other words, the robot adjusts the challenge to how the player moves around the robot when playing.

The behavior of the player is evaluated through a continuous registration of the players position and orientation of the body, which is inferred from 2D laser range measurements as explained in [16]. To detect persons the robot rely on the scans from the laser range finder using the leg detection algorithm presented in [17]. The algorithm is further supported by a Kalman filter for tracking and estimation of the person pose [16]. A more detailed description of the CBR database implementation can be found in [18].

### 3.3 Adaptive Robot Motion

The robot's navigation system is modeled using a person centered potential field, where the robot seeks towards the lowest values using a gradient descent. The potential field is calculated by the weighted sum of four Gaussian distributions of which one is negated. The covariance of the distributions are used to adapt the potential field according to *PSI*. When the player is inexperienced, the *PSI* values registered for a player will be closer to 0 and the robot will try to approach the player so he/she can hand over a ball to the robot. In the extreme case with *PSI*$= 0$ (an unskilled player), the potential field will like look Figure C.3, and the robot will enter the dark blue space right in front of the player. On the other hand, if the player is skilled, *PSI* will be closer to 1 and potential file will look like illustrated in Figure C.4. The robot will try to avoid the player by moving towards the dark blue area away from the player, making it more difficult for the player to hand over the ball.

Since the potential field is person centered, it moves with the player. If e.g. a skilled player starts moving towards the robot, the robot will eventually be in the yellow or red area in front of the player. The result is that the robot will start moving backwards towards the dark blue area, thus avoiding the player.

## 4 Experiments

A series of experiments have been designed to demonstrate the following features of the implemented system:

1. The learning capability of the CBR database. The generation of cases in the database.
2. Adaptiveness of the system. How the CBR database adapts to the skill of a player.
3. The estimate of the *PSI* of a player.

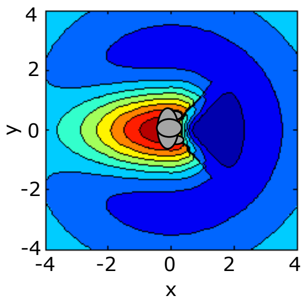


Figure C.3: The player's skill $PSI = 0$, and the robot will seek the dark blue area in front of the player

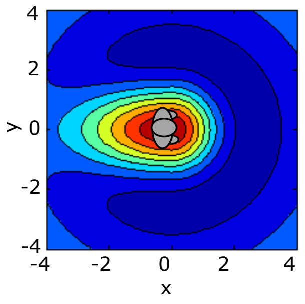


Figure C.4: The player's skill $PSI = 1$, and the robot will seek the dark blue area a bit away from the player

4. Adaptive Navigation. How the motion of the robot depends on the player skill.

5. Effect of Learning Rate. How the learning rate affect the *PSI*.

## 4.1 The learning capability of the CBR database.

This experiment should show that cases are actually created in the database and that these cases reflect the behavior of the player. To avoid a huge amount of repetitive playing time, simulations using the Player/Stage environment have been used to train the database. Using an empty CBR database, first the database is trained by a skilled player. Afterwards, a new database is created which is trained by an unskilled player.

### 4.2 Adaptiveness of the System

This experiment have been done using a combination of simulations and real world experiments. The two databases from the former experiment are used in a real world setting, where a test person is playing against the system. To show the system is capable of adapting to a new situation, an unskilled player plays with the system trained for a skilled player and vice versa. The average value of *PSI* in the whole database is logged continuously during the experiments.

### 4.3 Estimate of *PSI*

To show that the robot is able to estimate the player *PSI*, and hereafter adapt the motion accordingly, the trained databases are used again in the real world setting. This time the motion of both the person and the robot are recorded.

### 4.4 Effect of Learning Rate

A central parameter of the game algorithm, is the learning rate $L$ which is a numeric value in the interval $0 - 100$ used to control how fast *PSI* should adjust the estimate of the player's skill *PSI*. A simulation has been designed, such that in the first $10$ games, the player needs $4$ evaluation periods before he/she manages to hand the ball back to the robot. The effect should be that most of the cases in the database have a relatively low *PSI* value reflecting an unskilled player. In the last $40$ games, the simulation has been changed so the player hands back the ball to the robot within one evaluation period representing the behavior of a skilled player. To demonstrate the effect of changing the learning rate, the same simulation setup has been tried with the learning rate set to $0, 20, 40, 60, 80$ and $100$.

### 4.5 The robot platform

The robotic platform, which forms the basis of the experiment, is shown in Figure C.5. The robot is FESTO and is called Robotino. The robot is equipped with a head having 126 red diodes (see Figure C.5) which enables it to express different emotions. The robot is 1 meter high, and has mounted an URG-04LX line scan laser placed $35cm$ above ground level, scanning 220 degrees in front of the robot. In order to get feedback from the test person and find out when the robot has the ball, a cup with an on/off switch in the bottom, has been placed just below the robot's head, $75cm$ above ground level. The software framework Player [19] is installed on the platform and used for control of the robot and implementation of the CBR system.

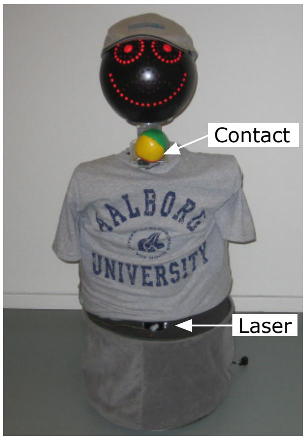


Figure C.5: The modified FESTO Robotino robotic platform.

## 5 Results

### 5.1 Trag the CBR System

Figure C.6 shows a plot of the CBR database, when the robot has been trained by an unskilled player. The position of the robot is $(0, 0)$. Each case in the database is represented by a short vector extending from a black dot in the figure. The color of the vector represents the value of *PSI* for the corresponding case. The pose and position of the player is represented by the corresponding position of the dot and angle of the vector. In Figure C.6, most vectors are in the color span between blue and green which represents *PSI* values between $0$ - $0.5$, and the average of all *PSI* values is $0.25$. This *PSI* range is as expected for an unskilled player, and it shows that the CBR system is capable of being trained for an unskilled player. Furthermore it can be seen that the database the is more densely populated closer to the robot. This is also expected, since a player will start off at a random direction away from the robot and will always move towards a point just in front of the robot.

Similar results have been obtained, when training with a skilled player. Here, most vectors are in the color span between green and red which represent *PSI* values between $0.5$ - $1$. This is expected for a skilled player and the average *PSI* of the whole database is $0.74$

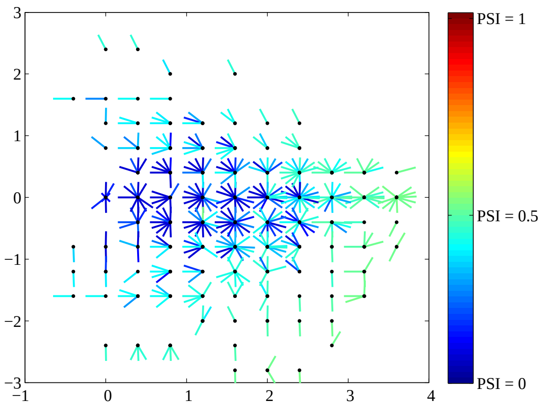


Figure C.6: A plot of the trained CBR database for an unskilled player. Each vector represents a case in the database using the features pose and position. The color of the vector denotes the *PSI* value using the color scale to the right. The robot is positioned in $(0, 0)$

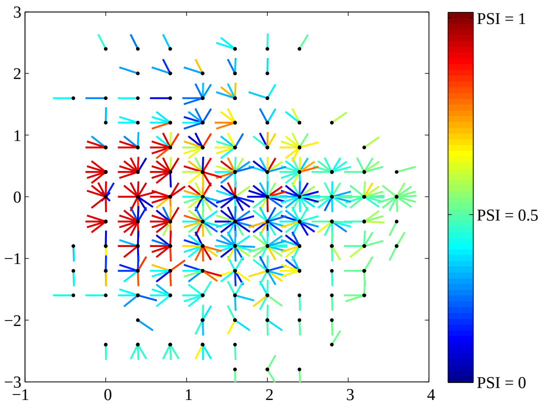


Figure C.7: A plot of the trained CBR database in Figure C.6 after the player has started to become good. The robot is positioned in $(0, 0)$.

## 5.2 Adaptiveness of the System

Using the trained database in Figure C.6, a skilled player is set to play in the real world. This makes the CBR database turn into the one shown in Figure C.7. It can be seen that the database has adapted to the player's skill, and has started to contain higher *PSI* values. Especially in the areas close to the robot, the *PSI* values have changed which is expected as most case updates happens here.

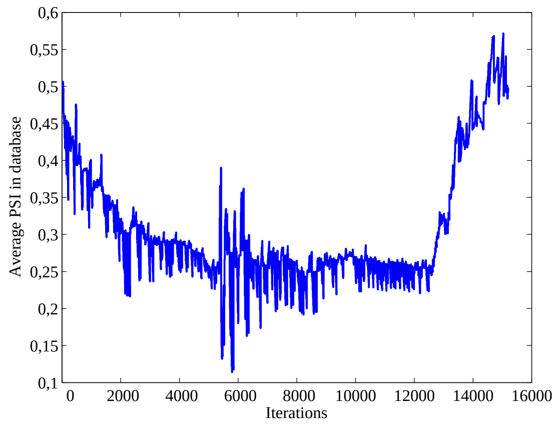


Figure C.8: The development of the average value of all *PSI* in the database after each iteration (each look up in the database). Initially *PSI* is around $0.5$ and the game is played by an unskilled player. The player slowly improves his skills and the average value of *PSI* increases correspondingly.

In Figure C.8 the development of the average value of *PSI* in the database can be seen. The value is saved for each lookup in the database, i.e. each time Line 20 in Algorithm I is passed. Initially the value is slightly less than $0.5$. After the database has been trained by an unskilled player for a while (after around $12000$ iterations) the average value has stabilized around $PSI = 0.25$. Now a skilled player starts playing, and it can be seen that the average value starts to increase rapidly, as expected. The noise on the figure is caused by individual trajectory differences.

A similar result has been obtained when starting with the system, which was trained by a skilled player, and played in the real world by a unskilled player. Here the database values are adapted from relative high values to lower values. This could be the case if a player starts to have more severe physical disabilities caused by e.g. a stroke.

## 5.3 Adaptive Navigation

Figure C.9 shows the trajectory of a person and for the robot, for a game where the robot has been trained by a skilled player. The person starts from the right side, and goes to pick up the ball. Hereafter the robot and person moves away from each other. When the game starts, the robot approaches slowly because it is far away. But as soon the person comes too close (when the trajectory is orange), the robot starts to move away. The player then tries to cheat the robot by moving sideways. This is a behavior the robot has not learned yet, and therefore it lets the person approach a bit more.

Figure C.10 shows the trajectory of a person and a robot for a game where the robot

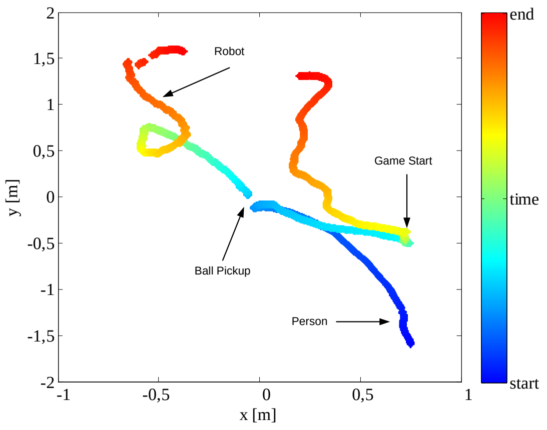


Figure C.9: The trajectory of a person and the robot for a game, where the robot has been trained by a skilled player. The color of the trajectory defines the time. Blue is the beginning and red is the end of the game.

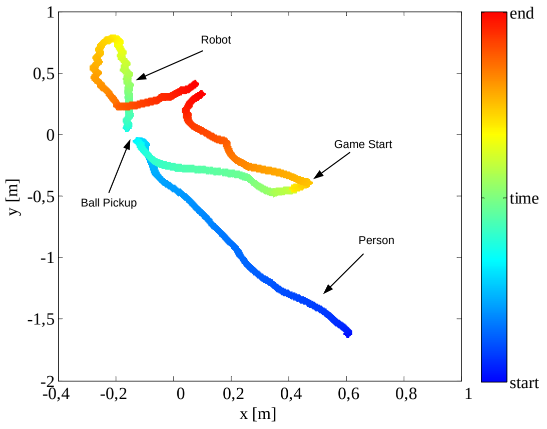


Figure C.10: The trajectory of a person and the robot for a game, where the robot has been trained by an unskilled player. The color of the trajectory defines the time. Blue is the beginning and red is the end.

has been trained by an unskilled player. The person starts from the lower right corner, and goes to pick up the ball. Hereafter the robot and person moves away from each other. Towards the end of the game, the robot approaches the person to make it easier for the player to hand back the ball. These two experiments show that the system is able to estimate the *PSI* correctly and navigate accordingly.

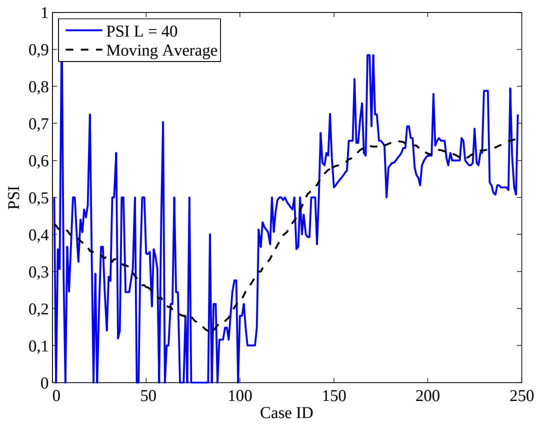


Figure C.11: Shows the development of the player skill *PSI* in a simulated set of $50$ games with one player using a learning rate on $L = 40$

## 5.4 Effect of Learning Rate

Figure C.11, shows the *PSI* values in the database for simulation set of $50$ games with one player using a learning rate set to $L = 40$. In total, $249$ cases are stored in the CBR database. Because the cases in the database are stored in the order they were observed, two consecutive cases do not necessarily have anything to do with each other and large fluctuations occur. A moving average gives an overview over what is happening, and as can be seen from the figure, the values of *PSI* decreases due to a gradually better trained database. Then, around case $100$, there is a sudden increase in *PSI*. This corresponds to the time when the player changes behavior from being unskilled to skilled and manage to hand the ball back in one evaluation period.

Table C.1[1], shows the same scenario with a learning rate set to $L = \{0, 20, 40, 60, 80, 100\}$. As $L$ increases, the deviation also increases which is expected. When the learning rate is $100$, *PSI* is adjusted with every little change of player behavior. The fluctuations of *PSI* becomes high, which makes the game algorithm too varying to be usable. On the other hand, *PSI* stays constant at $0.5$ when the learning rate is equal to $0$. The robot simply does not learn from its experience, and the game algorithm will never adapt the challenge to the player. The development of *PSI* using a learning rate of $0$ and $100$ is illustrated in Figure C.12. It has been chosen to use a learning rate of $L = 40$ for all experiments, since this value gives an adequate balance of adaptability and stability.

[1]This reference was wrong in the original paper. It has been corrected here.

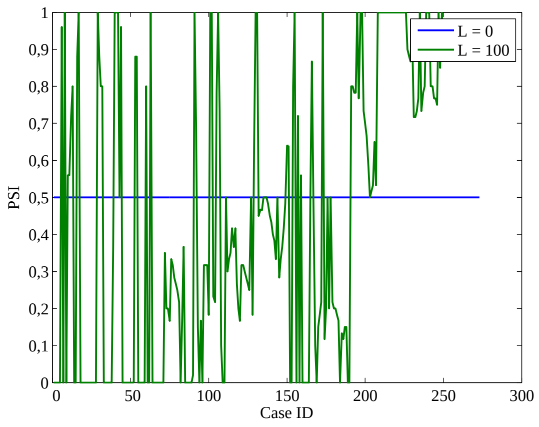


Figure C.12: Shows the development of the player skill *PSI* in a simulated set of 50 games with one player using a learning rate on $L = 0$ and $L = 100$

| $L$ | n | $\sigma$ |
|---|---|---|
| 0 | 273 | 0.00 |
| 20 | 248 | 0.18 |
| 40 | 249 | 0.23 |
| 60 | 259 | 0.27 |
| 80 | 230 | 0.29 |
| 100 | 251 | 0.38 |

Table C.1: The table shows the development of *PSI* for different learning rates $(L)$ with respect to the number of cases in the database $(n)$ and the standard deviation $\sigma$

.

## 6 Conclusion

In this paper, we have presented the concept of a robot based game. The function of the game is to increase the health of the players by motivating them to do a regular amount of physical exercise in a fun and social manner. The fact that the robot autonomously initiates the game, is a major difference from similar types of technology driven approaches e.g. using Nintendo Wii which has been successfully applied in a nursing home setting. The robot game is based on a simplified pursuit and evasion scenario, where the player should try to hand over a ball to the robot while the robot should try to avoid receiving the ball. Changing the behavior of the user through the use of technology is related to the term Persuasive Technology which is explained in the first section of the paper. The term Flow is often used to describe an ideal user experience in games. As described, one of the primary requisites of Flow is to provide an appropriate relation between game challenge and the user's skills. Based on this fact, a game algorithm has been designed and it is outlined how the algorithm works. The algorithm is implemented in a physical robot and the game is validated in simulation and through a practical lab experiment setup. Trajectories of the player and the robot in the lab experiment are documented along with plots of the CBR database which form the basis of the learning algorithm. The experiments document the learning capability of the CBR database and the adaptiveness of the system. It also shows how the system estimates the skill of the player and how it adapts depending on the learning rate parameter.

The next step will be to do a real world experiment at an activity center for elderly with the goal of measuring the user feedback and experience. Also we will work on enhancing the game, so the robot game is based on multiple players which will strengthen the social aspect of the game.

## References

[1] H. Zlotnik, Ed., *World Population Prospects - The 2004 Revision, Highlights*. United Nations, Population Division/DESA at www.unpopulation.org., 2005.

[2] A. Alzheimer's Disease International, "The prevalence of dementia worldwide," Alzheimer's Disease International, The International Federation of Alzheimer's Disease and Related Disorders Societies, Inc., Tech. Rep., 2008. [Online]. Available: http://www.alz.co.uk/adi/pdf/prevalence.pdf

[3] W. IJsselsteijn, H. H. Nap, Y. de Kort, and K. Poels, "Digital game design for elderly users," *Proceedings of the 2007 Conference on Future Play*, vol. Future Play '07, pp. 17–22, Toronto, Canada, November 14 - 17, 2007.

[4] G. Wendel-Vos, A. Schuit, E. Feskens, H. Boshuizen, W. Verschuren, W. Saris, and D. Kromhout, "Physical activity and stroke. a meta-analysis of observational data." *International Journal of Epidemiol.*, vol. 33(4), pp. 787–98, 2004 May 27.

[5] D. K. R. Fox, "The influence of physical activity on mental well being," *Public Health Nutrition*, vol. 2, p. 411, 1999.

[6] R. Brown, H. Sugarman, and A. Burstin, "Use of the nintendo wii fit for the treatment of balance problems in an elderly patient with stroke: A case report," *International Journal of Rehabilitation Research, Proceedings of the 10th Congress of the European Federation for Research in Rehabilitation*, vol. 32, p. 109, 2009.

[7] C. Neufeldt, "Wii play with elderly people," *international reports on socio-informatics*, vol. 6, 2009.

[8] C. W. Reynolds, “Competition, coevolution and the game of tag,” in *Artificial Life IV*, 1994.

[9] B. Fogg, *Persuasive Technology. Using Computers to Change What We Think and Do*. Morgan Kaufmann, 2003.

[10] B. J. Fogg, “Captology. the study of computers as persuasive technologies,” *Proceedings of the CHI 97, Extended abstracts on Human factors in computing systems*, vol. New York: AMC Press, p. 129, 1997.

[11] M. Csikszentmihalyi, *Beyond boredom and anxiety*. Jossey-Bass Publishers, 1975.

[12] M. E. Pertou and S. D. Iversen, “Persuasivt design i retorisk perspektiv,” *RhetoricaScandinavica*, vol. 49/50, 2009.

[13] F. L. Jensen, “Flow & læringsspil,” Master’s thesis, Aalborg University, 2006.

[14] J. Kolodner, *Case-based Reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[15] I. Jurisica and J. Glasgow, “Applying case-based reasoning to control in robotics,” in *3rd Robotics and Knowledge-Based Systems Workshop, St. Hubert Quebec*, 1995.

[16] M. Svenstrup, S. T. Hansen, H. J. Andersen, and T. Bak, “Pose estimation and adaptive robot behaviour for human-robot interaction,” in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation. ICRA 2009.*, Kobe, Japan, May 2009, pp. –.

[17] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, “Fast line, arc/circle and leg detection from laser scan data in a player driver,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005.*, 18-22 April 2005, pp. 3930–3935.

[18] S. T. Hansen, M. Svenstrup, H. J. Andersen, and T. Bak, “Adaptive human aware navigation based on motion pattern analysis,” in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, Toyama, Japan, September-October 2009.

[19] T. Collett, B. A. MacDonald, and B. P. Gerkey, “Player 2.0: Toward a practical robot programming framework,” in *In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, C. Sammut, Ed., Sydney, Australia, December 2005, http://playerstage.sourceforge.net.

# Paper D

**Trajectory Planning for Robots in Dynamic Human Environments**

Mikael Svenstrup, Thomas Bak and Hans Jørgen Andersen

This paper is published in:
Proceedings of the 2010 International Conference Intelligent Robots and Systems, IROS 2010

**Abstract**

This paper presents a trajectory planning algorithm for a robot operating in dynamic human environments. Environments such as pedestrian streets, hospital corridors, train stations or airports. We formulate the problem as planning a minimal cost trajectory through a potential field, defined from the perceived position and motion of persons in the environment.

A Rapidly-exploring Random Tree (RRT) algorithm is proposed as a solution to the planning problem, and a new method for selecting the best trajectory in the RRT, according to the cost of traversing a potential field, is presented. The RRT expansion is enhanced to account for the kinodynamic robot constraints by using a robot motion model and a controller to add a reachable vertex to the tree.

Instead of executing a whole trajectory, when planned, the algorithm uses a Model Predictive Control (MPC) approach, where only a short segment of the trajectory is executed while a new iteration of the RRT is computed.

The planning algorithm is demonstrated in a simulated pedestrian street environment.

## 1 Introduction

As robots integrate further into our living environments, it becomes necessary to develop methods that enable them to navigate in a safe, reliable, comfortable and natural way around humans.

One way to view this problem is to see humans as dynamic obstacles that have social zones, which must be respected. Such zones can be represented by potential fields [1, 2]. The navigation problem can then be addressed as a trajectory planning problem for dynamic environments with a potential field representation. Given the fast dynamic nature of the problem, robotic kinodynamic and nonholonomic constraints must also be considered.

In the recent decade sampling based planning methods have proved successful for trajectory planning [3]. They do not guarantee an optimal solution, but are often good at finding solutions in complex and high dimensional problems. Specifically for kinodynamic systems Rapidly-exploring Random Trees (RRT's), where a tree with nodes correspond to connected configurations (vertices) of the robot trajectory, has received attention [4].

Various approaches improving the basic RRT algorithm have been investigated. In [5], a dynamic model of the robot and a cost function is used to expand and prune the nodes of the tree. A Model Predictive Control (MPC) approach is taken, where only a small part of the trajectory is executed, while a new trajectory is calculated. When expanding a vertex, a random vertex and a random control input is chosen.

In [6], an approach for better choices of vertices to expand, is proposed. It is based on a reachable set of configurations for each vertex.

It is often desirable to run the planning algorithm in real time, hence requiring bounded solution time. One approach is to use anytime algorithms, which initially find a quick suboptimal solution, and then keep improving the solution until time runs out [7].

Methods for incorporating dynamic environments, have also been investigated. Solutions include extending the configuration space with a time dimension ($\mathcal{C} - \mathcal{T}$ space), in which the obstacles are static [8], as well as pruning and rebuilding the tree when changes occur [9, 10].

All these result only focus on avoiding collisions with obstacles. However, there have been no attempts to navigate through a human crowd taking into account the dynamics of the environment and at the same time generate comfortable and natural trajectories around the humans. E. Hall [11] has analysed how people position themselves socially relative to each other. He divides the area around a person into four zones (public, social, personal and intimate). These zones can be used to plan how a robot should move to make the motion natural and comfortable.

In this paper will formulate the problem of navigating through a dynamic human environment, as planning a trajectory through a potential field. The overall mission of the robot is, to move forward through the environment with a desired average speed and direction, which can be set by a top level planner. This paper contributes by enhancing the basic RRT planning algorithm to accommodate for navigation in a potential field and take into account the kinodynamic constraints of the robot. The RRT is expanded using a control strategy, which ensures feasibility of the trajectory and a better coverage of the configuration space. The planning is done in $\mathcal{C} - \mathcal{T}$ space using an MPC scheme to incorporate the dynamics of the environment. To be able to run the algorithm on-line, the anytime concept is used to quickly generate a possible trajectory. The RRT keeps being improved until a new trajectory is required by the robot, which can happen at any time.

The trajectory planning problem is formulated in Section 2, and the algorithm for generating the trajectory is described in Section 3. Finally in Sections 4-5 the algorithm is demonstrated in an experiment, where the robot plans the trajectory through a simulated pedestrian street.

## 2 Trajectory Generation Problem

### 2.1 Robot Dynamics

The robot is modelled as a unicycle type robot, i.e. like a Pioneer, an iRobot Create or a Segway. A good motion model for the robot is necessary because it operates in dynamic environments, where even small deviations from the expected trajectory may result in collisions. So instead of using a purely kinematic robot model of the robot, it is modelled as a dynamical system, with accelerations as input. This describes the physics better, since acceleration and applied force are proportional. This dynamical model can be described by the five states:

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \begin{matrix} \rightarrow \text{x position} \\ \rightarrow \text{y position} \\ \rightarrow \text{linear velocity} \\ \rightarrow \text{rotation angle} \\ \rightarrow \text{rotational velocity} \end{matrix} \tag{D.1}$$

The differential equation governing the robot behaviour is:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\theta}(t) \\ \ddot{\theta}(t) \end{bmatrix} \\ &= \begin{bmatrix} v(t)\cos(\theta(t)) \\ v(t)\sin(\theta(t)) \\ u_v(t) \\ \dot{\theta}(t) \\ u_\theta(t) \end{bmatrix} = \begin{bmatrix} x_3(t)\cos(x_4(t)) \\ x_3(t)\sin(x_4(t)) \\ u_1(t) \\ x_5(t) \\ u_2(t) \end{bmatrix} \end{aligned} \tag{D.2}$$

where $u_1 = u_v$ is the linear acceleration input and $u_2 = u_\theta$ is the rotational acceleration input.

Without loss of generality the starting time can be set to 0, and the trajectory is then calculated as:

$$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t \boldsymbol{f}\left(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right) d\tau \tag{D.3}$$

## 2.2 Dynamic Potential Field

The value of the potential field, denoted $G$, at a point in the environment is calculated as a sum of the cost associated with three different aspects:

1. A cost related to the robots position in the environment without obstacles. For example high costs might be assigned close to the edges.

2. A cost associated with the robot position relative to humans in the area.

3. A cost rewarding moving towards a goal.

The combined cost can be written as:

$$G(t) = g_1(\boldsymbol{x}(t))) + g_2(\boldsymbol{x}(t), \mathcal{P}(t)) + g_3(\boldsymbol{x}(t)) \quad . \tag{D.4}$$

$\mathcal{P}(t)$ is a matrix containing the position and orientation of persons in the environment at the given time. $g_1(\boldsymbol{x})$, $g_2(\boldsymbol{x})$ and $g_3(\boldsymbol{x})$ are the three cost functions. They are further described below.

### Cost related to environment

This cost function is currently designed for non agoraphobic behaviour of the robot, i.e. in open spaces, such as a pedestrian street. It has the shape of a valley, such that it is more expensive to go towards the sides, but cheap to stay in the middle:

$$g_1(\boldsymbol{x}(t)) = c_y y^2(t) \tag{D.5}$$

where $c_y$ is a constant determining how much the robot is drawn towards the middle.

### Cost of proximity to humans

This is not a straightforward calculation, and for more detail, see [2]. The shape of the potential field is related to how humans position themselves around others, and is based on Hall's proxemic distances [11]. For example the potential is lower in front of the person than behind, because it is more comfortable to have other persons, where you can see them.

Fig. D.1 shows a potential field around a person. The person stands in the point $(0, 0)$ and is looking to the left. A robot should try to move towards the lower parts of the potential function, i.e. towards the dark blue areas, and avoid the red area. The formula for calculating the potential around one person is a summation of four normalized bi-variate Gaussian distributions:

$$g_2(\boldsymbol{x}_{1:2}) = \sum_{k=1}^{4} c_k \exp(-\frac{1}{2}[\boldsymbol{x}_{1:2} - \mathbf{0}]^T \Sigma_k^{-1} [\boldsymbol{x}_{1:2} - \mathbf{0}]) \tag{D.6}$$

where $c_k$ are a normalizing constants, $\boldsymbol{x}_{1:2}$ are the first two states of the robot state, i.e. the position relative to the person, where the cost function is evaluated. $\mathbf{0}$ is the position of the person, in this case the origin, and $\Sigma_k$ are the covariances of each of the Gaussian distributions. The covariances are adjusted according to the orientation of the person. The total cost, $g_2$ is a summation over all of the persons in the area.

### Cost of end point in trajectory

The cost at the end point penalizes if the robot does not move forward, and if the robot orientation is not in a forward direction. An exponential function is used to penalize the

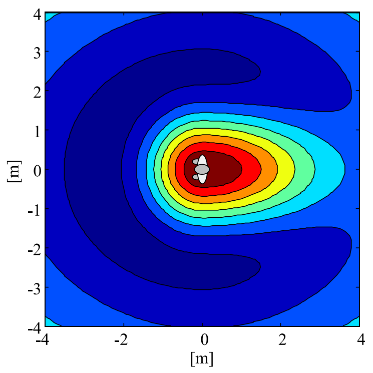


Figure D.1: Potential field around a person standing at $(0, 0)$ and looking to the left. The robot should try to get towards the lower points, i.e. the dark blue areas. The size of the person in this figure is approximate equal to a normal human being.

position. It is set up such that short distances are penalized much, while it is close to the same value for larger distances, i.e. it does not change much if the robot goes 19 or 20 meters from its starting position.

$$g_3(\boldsymbol{x}(t)) = c_{e1} \exp(c_{e2}(x(t) - \tilde{x}(0))) + c_\theta \theta^4(t) \quad , \tag{D.7}$$

where $c_{(\cdot)}$ are scaling constants and $\tilde{x}(0)$ is the desired position at $t = 0$. The reason that $\theta$ is raised to the fourth, is to keep the term closer to zero in a larger neighbourhood of the origin. This means that the robot will almost not be penalized for small turns. On the other hand larger turns, like going the wrong way, will be penalized more.

## 2.3 Minimization Problem

Given the above cost functions a potential landscape may be formed. Fig. D.2 illustrates an example of a pedestrian street landscape with five persons. The robot is initially positioned at position $(2, 0)$ and has to move to the right. The area is bounded to be $20m$ wide, i.e. $10m$ to each side of the robot from the initial position. Examples of three different randomly chosen trajectories are shown in the figure.

At a first glance it looks like all three trajectories would run into at least one human, but since the persons move while the robot advances along the trajectory, this might not be the case. Conversely the robot may also run into a person, who was not originally on the path. Therefor it is important to take into account the dynamics of the obstacles (i.e. the humans), when planning trajectories.

If the current time is $t = 0$, the planning problem can be posed as follows. Given an initial robot state $\boldsymbol{x}_0$, and trajectory information for all persons until the given time

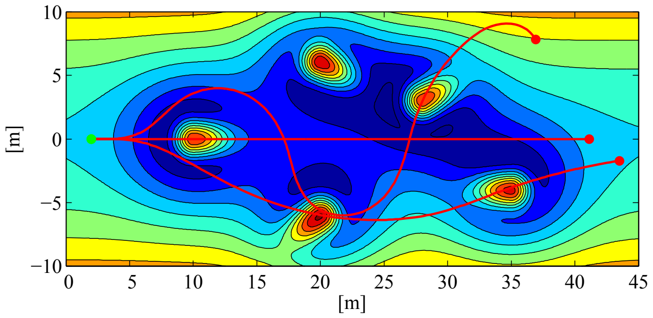


Figure D.2: Person potential field landscape, which the robot has to move through. The robot starting point is the green dot at the point $(2, 0)$. Three examples of potential robot trajectories are shown.

$\tilde{\mathcal{P}}_{start:0}$. Determine the control input $\tilde{\boldsymbol{u}}_{0:T}$, which minimizes the cost of traversing the potential field, subject to the dynamical robot model constraints:

$$\begin{aligned}
minimize \quad & I(\tilde{\boldsymbol{u}}_{0:T}) = \\
& \int_0^T [g_1(\boldsymbol{x}(t)) + g_2(\boldsymbol{x}(t), \mathcal{P}(t))]\, dt + g_3(\boldsymbol{x}(T)) \\
s.t. \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\,(\boldsymbol{x}(t), \boldsymbol{u}_t) \\
where \quad & g_1(\boldsymbol{x}(t)) = c_y x_2(t)^2 \\
& g_2(\boldsymbol{x}(t), \mathcal{P}(t)) = \\
& \sum_{j=1}^{p} \sum_{k=1}^{4} c_k \exp(-\frac{1}{2}[\boldsymbol{x_{1:2}} - \boldsymbol{\mu}_j]^T \Sigma_{j,k}^{-1} [\boldsymbol{x_{1:2}} - \boldsymbol{\mu}_j]) \\
& g_3(\boldsymbol{x}(T)) = c_{e1} \exp(c_{e2}(x_1(T) - x_1(0))) + c_\theta x_4^4(T),
\end{aligned} \tag{D.8}$$

were $\boldsymbol{x_{1:2}} = [x_1(t), x_2(t)]$ is the position of the robot at time $t$, $\tilde{\boldsymbol{u}}_{0:T}$ is the discrete input sequence to the robot. $T$ is the ending time horizon of the trajectory, $g_x(\cdot)$ are cost functions and $p$ is the number of persons in the area. The position and orientation of all persons at time $t$ is given by $\mathcal{P}(t)$ and $\boldsymbol{\mu}_j$ is the center of the $j$-th person at a given time.

To be able to calculate the cost of a trajectory according to Eq. (D.8), only the person trajectories remains to be defined. A simple model is that the person will continue with the same speed and direction [12]. More advanced human motion models could be used without changing the planning algorithm, but it is outside the scope of this paper to derive a complex human motion model.

## 3 RRT Based Trajectory Planning

The structure of the planning algorithm can be seen in Fig.D.3. The idea is that while a trajectory is executed, a new is calculated on-line. Input to the trajectory planner is the previous best trajectory, the person trajectory estimates, and the dynamic model of the robot.

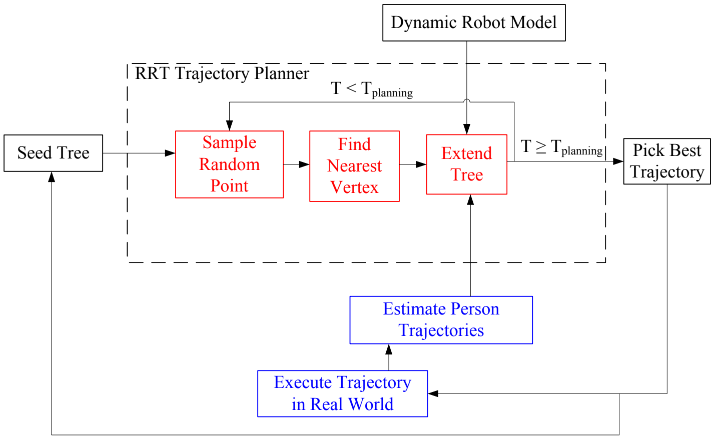


Figure D.3: The overall structure of the trajectory generator. The blue real world part and the red trajectory planning part are executed simultaneously.

The minimization problem stated in Eq. (D.8) is addressed by RRT's. A standard RRT algorithm is shown in Algorithm 1, where the lines $4, 5, 6$ correspond to the three blocks in the larger *RRT Trajectory Planner* box in Fig. D.3.

The method presented here differs from the standard RRT in lines $1, 3, 6, 9$, which are marked red. Furthermore, between line $6$ and $7$, node pruning is introduced. Since an MPC scheme is used, only a small portion of the planned trajectory is executed, while the planner is restarted to plan a new trajectory on-line. When the small portion has been executed, the planner has an updated trajectory ready. To facilitate this, the stopping condition in line $3$ is changed. When a the robot needs a new trajectory, or when certain maximum number of vertices have been extended, the RRT is stopped. Even though the robot only executes a small part of the trajectory, the rest of the trajectory should still be valid. Therefore, in line $1$, the tree is seeded with the remaining trajectory.

In line $9$ the trajectory with the least cost is returned, instead of returning the trajectory to the newest vertex. The tree extension function and the pruning method are described below.

**Algorithm 4** Standard RRT (see [13])

```
RRTmain()
1: Tree = q.start
2: q.new = q.start
3: while Dist(q.new , q.goal) < ErrTolerance do
4:     q.target = SampleTarget()
5:     q.nearest = NearestVertex(Tree , q.target)
6:     q.new = ExtendTowards(q.nearest,q.target)
7:     Tree.add(q.new)
8: end while
9: return Trajectory(Tree,q.new)
SampleTarget()
1: if Rand() < GoalSamplingProb then
2:     return q.goal
3: else
4:     return RandomConfiguration()
5: end if
```

## 3.1 RRT Control Input Sampling

When working with nonholonomic kinodynamic constrained systems, it is not straightforward to expand the tree towards a newly sampled point in the configuration space (line 6 in Algorithm 1). It is a whole motion planning problem in itself to find inputs, that drive the robot towards a given point [14]. The algorithm proposed here uses a controller to turn the robot towards the sampled point and to keep a desired speed. A velocity controller is set to control the speed towards an average speed around a reference velocity. The probabilistic completeness of RRT's in general, is ensured by the randomness of the input. So to maintain this randomness in the input signals, a random value sampled from a Gaussian distribution is added to the controller input. The velocity controller is implemented as a standard proportional controller. The rotation angle is a second order system with $\theta$ and $\dot{\theta}$ as states, and therefore a state space controller is used for control of the orientation. The control input can be written as:

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_v(\mu_v - v(t)) \\ k_{\theta 1}(\phi_{point} - \theta(t)) - k_{\theta 2}\dot{\theta} \end{bmatrix} + \begin{bmatrix} \mathcal{N}(0, \sigma_v) \\ \mathcal{N}(0, \sigma_\theta) \end{bmatrix}. \tag{D.9}$$

$\mu_v$ is the desired average speed of the robot and $\phi_{point}$ is the angle towards the sampled point. $k_{(\cdot)}$ are controller constants and $\sigma_v, \sigma_\theta$ are the standard deviations of the added Gaussian distributed input.

The new vertex to be added to the tree is now found by using the dynamic robot motion model and the controller to simulate the trajectory one time step. This ensures that the added vertex will be reachable.

## 3.2 Tree Pruning and Trajectory Selection

A simple pruning scheme, based on several different properties of a node, is used. If the vertex corresponding to the node ends up in a place where the potential field has a value above a specific threshold, then the node is not added to the tree. Furthermore a node is pruned if $|\theta(t)| > \frac{\pi}{2}$, which means that the robot is on the way back again, or if the simulated trajectory goes out of bounds of the environment. It is not desirable to let the tree grow too far in time, since the processing power is much better spend on the near future, because of the uncertainty of person positions further into the future. Therefore the node is also pruned if the time taken to reach the node is above a given threshold. Finally, instead of returning the trajectory to the vertex of the last added node, the trajectory with the lowest cost (calculated from Eq. (D.8)), is returned. But to avoid the risk of selecting a node, which is not very far in time, all nodes with a small time are thrown away before selecting the best node.

The final algorithm is shown in Algorithm 2.

**Algorithm 5** Modified RRT for human environments

```
RRTmain()
 1: Tree = q.oldBestTrajectory
 2: while (Nnodes < maxNodes) and (t < tMax) do
 3:    q.target = SampleTarget()
 4:    q.nearest = NearestVertex(Tree , q.target)
 5:    q.new = CalculateControlInput(q.nearest,q.target)
 6:    if PruneNode(q.new) == false then
 7:       Tree.add(q.new)
 8:    end if
 9: end while
10: return BestTrajectory(Tree)
SampleTarget()
 1: if Rand < GoalSamplingProb then
 2:    return q.goal
 3: else
 4:    return RandomConfiguration()
 5: end if
```

## 4 Simulations

The above described algorithm is implemented, and demonstrated to work on a simulated pedestrian street, as shown in Fig. D.2. The experiments consist of two parts. First, the algorithm is applied on the environment shown in Fig. D.2. This will demonstrate that the algorithm is capable of planning a trajectory, which does not collide with any persons.

It will also demonstrate how the tree expands. The algorithm is compared to an algorithm where a random vertex and a random control input is chosen, as suggested in [4] and used to different extends in e.g. [5, 6]. Next, a simulated navigation through several randomly generated worlds is performed. This will demonstrate the robustness of the algorithm over time.

The following parameters for the potential field are used: $c_y = 0.1$, $c_{e1} = 20$, $c_{e2} = -0.1$, $c_\theta = 10$, and the parameters for the Gaussian distributions can be seen in [2]. The poles of the controllers, the reference velocity and the standard deviation of the velocity input, are the only other parameters to set. The poles have experimentally been determined, such that the robot has a relatively quick response, but the exact pole placement does not influence the trajectory generation much. The pole of the velocity controller is placed in $s = -2$, and both the poles of the rotational controller are placed in $s = -2$ as well. The standard deviation of the added random velocity input is set to $\sigma_v = 2\frac{m}{s}$ and the standard deviation of the rotational input is set to $\sigma_\theta = 0.5\frac{rad}{s}$. The reference velocity is set to $1.5\frac{m}{s}$, which is considered as a normal human walking speed.

## 4.1 Robustness Test

The robustness test is performed in $50$ different randomly generated environments, where the robot has to navigate forwards in one minute. With an average speed of $1.5\frac{m}{s}$, this corresponds to the robot moving approximately $90m$ ahead along the street. In each simulation the robot's initial state is:

$$\boldsymbol{x}_0 = [2\ 0\ 0\ 0\ 0]^T \tag{D.10}$$

First the motion of all the persons in the world are simulated. Initially a random number of persons (between $10$ and $20$) are placed randomly in the world. Their velocity is sampled randomly from a Gaussian distribution. The motion of each person is simulated as moving towards a goal $10m$ ahead of them. The goal position of the $y - axis$ of the street is sampled randomly, and will also change every few seconds. Additional Brownian motion is added to each person to include randomness of the motion. Over time new persons will enter at the end of the street according to a Poisson process. This means that at any given time, persons will appear and disappear at the ends of the street. Because of this randomness, the number of persons can differ from the initial number of persons, and ranges from $10$ to around $40$, which is different for each simulation.

At each time instant, the robot will only know the current position and velocity of each of the persons within a range of $45m$ in front of the robot, and it has no knowledge about where the persons will go in the future.

As it is a simulation, there is no real time performance issues, and nothing has been done to optimize the code for faster performance. So the tree is set to grow a fixed number of 2000 vertices at each iteration. The planning horizon is set to 20 seconds and at each iteration the robot executes 2 seconds of the trajectory, while a new trajectory is planned.

## 5 Results

An example of a grown RRT, with 2000 vertices, from the initial state can be seen in Fig. D.4. The simulated trajectories of the robot are the red lines, and the red dots are vertices of the tree. It is seen how the RRT is spread out to explore the configuration space, although only every 10th vertex is plotted to avoid clutter on the graph. Note that the persons are static at their initial position on the figure, and some trajectories seem to pass through persons. But in reality, the persons have moved when the robot passes the point of the trajectory. The best of the all trajectories, which is calculated using Eq. (D.8), is the green trajectory.

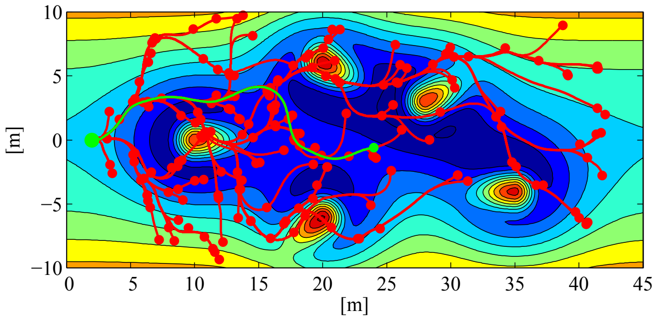


Figure D.4: An RRT for a robot starting at $(2, 0)$ and the task of moving forward through the human populated environment. Only every 10th vertex is shown to avoid clutter of the graph. The vertices are the red dots, and the lines are the simulated trajectories. The green trajectory is the least cost trajectory.

In Fig. D.5 a RRT has been run for the same environment, but by choosing a random vertex to expand, instead of the one closest to a random sampled point. The algorithm is a little faster, since it does not have to calculate distances to all vertices. But as can be seen, the tree does almost not expand over the configuration space.

An example of the tree after 2000 expansions, when choosing the nearest vertex, but using a random control input , is illustrated in Fig. D.6. The configuration space is not covered very well, since there are only two major branches of the tree, and the top area above the first person has not been explored at all. When comparing to Fig. D.4, it is clear that our control sampling method covers the configuration space much better.

In Fig. D.7 a typical scene from one of the 50 simulations can be seen. The blue dots are persons, and the arrows are velocity vectors, with a length proportional to the speed. The black star with the red arrow, is the robot position and orientation.

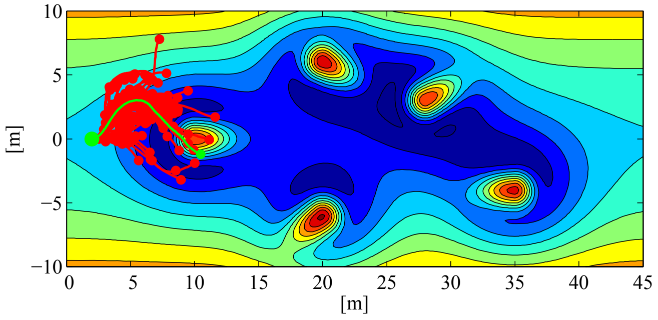


Figure D.5: The RRT after 2000 expansions, where a random vertex is chosen to be expanded. The tree expands very slow over the configuration space. The lowest cost trajectory is shown in green.

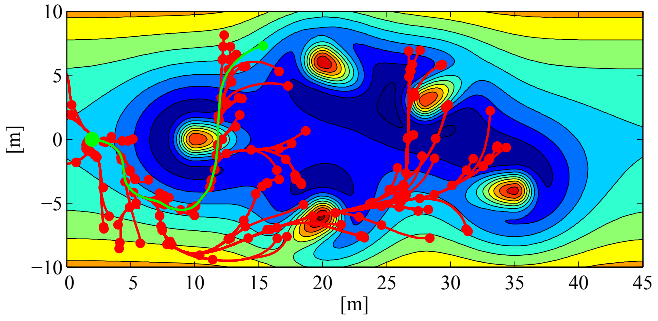


Figure D.6: An example of choosing random control input. The tree expands better than if choosing random nodes, but it still does not cover the configuration space very well. The lowest cost trajectory is shown in green.

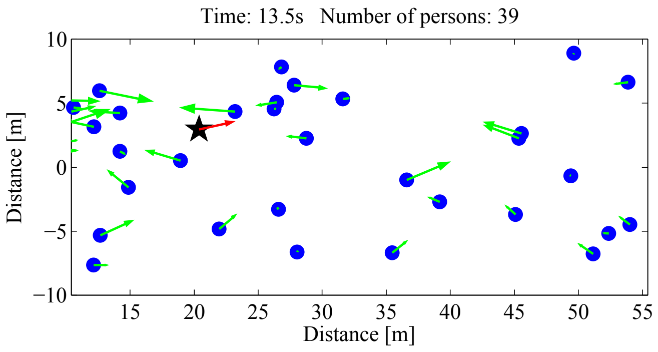


Figure D.7: A scene from one of the 50 simulations. The blue dots are persons, with their corresponding current velocity vectors. The black star is the robot.

In none of the 50 simulations the robot ran into a person. This demonstrates that the algorithm is robust enough to handle simulated human motion with changing goals and additional random motion, even though using the simple human motion model when planning. A few close passes in the combined 50 minute run, down the pedestrian street, are seen. But, as seen in Fig. D.8, the robot stays out of the personal zone of any of the persons in more than $97.5\%$ of the time, and out of the intimate zone (a distance closer than $0.45m$) $99.7\%$ of the time. This only occurs on 9 separate instances of the 50 minutes of driving.

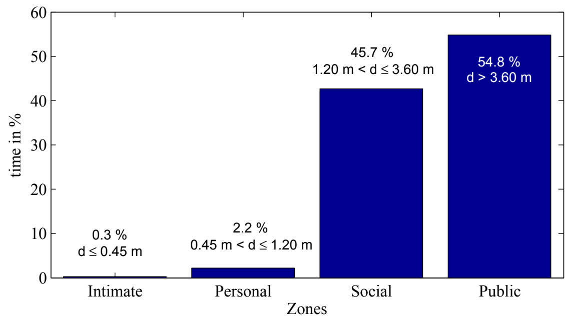


Figure D.8: The bar plot shows how large a part of the time the closest person to the robot has been in each zone. The robot should try to stay out of the personal and intimate zones. The distance interval for each zone, can also be seen in the plot.

Except in very densely populated environments, the model runs approximately one third of real time on a 2.0 $GHz$ CPU running MATLAB. This is considered to be reasonable, since no optimization for speed has been done.

In very dense environments, the planning takes longer, since many new added vertices, are pruned again, and hence more points has to be sampled before 2000 vertices are expanded. Additionally the more persons in the area, the longer it takes to evaluate the cost of traversing the potential field.

On average approximately $\frac{1}{4}$ of the sampled points led to an expansion of the tree and correspondingly $\frac{3}{4}$ led to a pruned node.

## 6 Conclusions

In this paper a new algorithm for trajectory planning for a kinodynamic constrained robot, has been described. The robot is navigating in a highly dynamic environment, which in this case is populated with humans. The algorithm is based on RRT's, but with a new trajectory selection method. The method enables the costs of traversing a potential field to be minimized, thereby supporting planning of comfortable and natural trajectories. Further, a new control input sampling strategy has been presented. This leads to better tree coverage over the configuration space, than if sampling e.g. a random control input. Together with a dynamic model of the robot, an MPC scheme is used to enable the planner to continuously plan a reachable trajectory on an on-line system.

The algorithm is challenged when the environments become very densely populated, but so are humans. Humans react by mutual adaptation and allowing violation of the social zones. This is not done here, where the robot takes on all the responsibility for finding a trajectory.

Potential future work include real life experiments, and incorporation of human to human motion correlation into the algorithm.

## References

[1] E. Sisbot, A. Clodic, L. Marin U., M. Fontmarty, L. Brethes, and R. Alami, “Implementing a human-aware robot system,” in *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006.*, 6-8 Sept. 2006, pp. 727–732.

[2] M. Svenstrup, S. T. Hansen, H. J. Andersen, and T. Bak, “Pose estimation and adaptive robot behaviour for human-robot interaction,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, Kobe, Japan, May 2009, pp. 3571–3576.

[3] S. LaValle, *Planning algorithms.* Cambridge Univ Pr, 2006.

[4] S. LaValle and J. Kuffner Jr, “Randomized kinodynamic planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, p. 378, 2001.

[5] A. Brooks, T. Kaupp, and A. Makarenko, “Randomised mpc-based motion-planning for mobile robot obstacle avoidance,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, May 2009, pp. 3962–3967.

[6] A. Shkolnik, M. Walter, and R. Tedrake, “Reachability-guided sampling for planning under differential constraints,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, May 2009, pp. 2859–2865.

[7] D. Ferguson and A. Stentz, “Anytime, dynamic planning in high-dimensional search spaces,” in *Proc. IEEE International Conference on Robotics and Automation*, 2007, pp. 1310–1315.

[8] J. van den Berg, “Path planning in dynamic environments,” Ph.D. dissertation, Ph. D. dissertation, Universiteit Utrecht, 2007.

[9] D. Ferguson, N. Kalra, and A. Stentz, “Replanning with rrts,” in *Proc. IEEE International Conference on Robotics and Automation ICRA 2006*, 2006, pp. 1243–1248.

[10] M. Zucker, J. Kuffner, and M. Branicky, “Multipartite rrts for rapid replanning in dynamic environments,” in *Proc. IEEE International Conference on Robotics and Automation*, 2007, pp. 1603–1609.

[11] E. T. Hall, “A system for the notation of proxemic behavior,” *American anthropologist*, vol. 65, no. 5, pp. 1003–1026, 1963.

[12] A. Bruce and G. Gordon, “Better motion prediction for people-tracking,” in *Robotics and Automation, 2004. ICRA ’04. IEEE International Conference on*, April 2004.

[13] D. Ferguson and A. Stentz, “Anytime rrts,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[14] B. Siciliano and O. Khatib, *Handbook of Robotics*. Springer-Verlag, Heidelberg, 2008.

# Paper E

## Minimising Computational Complexity of the RRT Algorithm

**A Practical Approach**

Mikael Svenstrup, Thomas Bak and Hans Jørgen Andersen

This paper is accepted for publication in:
Proceedings of the 2011 International Conference Robotics and Automation,
ICRA 2011

**Abstract**

Sampling based techniques for robot motion planning have become more widespread during the last decade. The algorithms however, still struggle with for example narrow passages in the configuration space and suffer from high number of necessary samples, especially in higher dimensions. A widely used method is Rapidly-exploring Random Trees (RRT's). One problem with this method is the nearest neighbour search time, which grows significantly when adding a large number of vertices. We propose an algorithm which decreases the computation time, such that more vertices can be added in the same amount of time to generate better trajectories. The algorithm is based on subdividing the configuration space into boxes, where only specific boxes needs to be searched to find the nearest neighbour. It is shown that the computational complexity is lowered from a theoretical point of view. The result is an algorithm that can provide better trajectories within a given time period, or alternatively compute trajectories faster. In simulation the algorithm is verified for a simple RRT implementation and in a more specific case where a robot has to plan a path through a human inhabited environment.

## 1 Introduction

Sampling based methods for motion and path planning have gained more interest during the last decade, as computer power has increased. Among the most widely used techniques are Rapidly-exploring Random Trees (RRT's)[1], which work by randomly expanding a tree with vertices over the configuration space to find a path from a start location to a goal location. RRT's have been used in various applications such as kinodynamic path planning [2], navigation for urban driving [3], coordination of robot motion [4], and humanoid robot motion planning [5]. The main strengths of an RRT are the ease of implementation, the ability to avoid obstacles and the applicability for systems with differential constraints. RRT algorithms do, however, suffer from the large number of vertices necessary in trees for large dimension problems and for problems with large configuration spaces including many obstacles. When adding a large number of vertices to a tree, especially the time it takes to find the nearest other vertex in each iteration is time consuming.

There are two ways to address this problem. One approach is to make a smarter algorithm, which needs fewer samples to obtain the goal. Another approach is to speed up the search for the nearest neighbour, such that more vertices can be added within the same time period. Most research has focused on the first problem, where most of the ideas come from trying to intelligently bias the exploration towards unexplored portions of the configuration space, such that the algorithm needs fewer samples to converge towards a goal [6, 7]. This can be done by either biasing the sampling distribution or improving the vertex extension operation. The nearest neighbour search typically rely on a naive brute force method, where the distance to all vertices are calculated, to find the nearest. This is mainly because it is by far the simplest way to do it. Not much research within the motion planning community has been done trying to optimise the speed of the nearest neighbour search. Similar problems have, however, been studied generally in the field of computational geometry under the name *range search problems* [8], and have applications in e.g. computer graphics and when querying large databases [9, 10, 11].

Good candidate algorithms for finding the nearest neighbour are found using a tree based structure to subdivide the configuration space. Algorithms include quadtrees, R-trees and $kd$-trees. Quadtrees and R-trees have a bad worst case performance, which may be why the $kd$-tree is the most widely used. A $kd$-tree works by recursively subdividing the space into two half spaces one dimension at the time (see [1, 12]). The query time complexity of a $kd$-tree is $\mathcal{O}(n^{1-\frac{1}{d}})$ (compared to $\mathcal{O}(n)$ for a brute force approach), where $n$ is number of vertices and $d$ is the number of dimensions. However, query time can be improved substantially for higher dimensional search spaces, if it is enough to find the approximate nearest neighbour (ANN) [13, 12], where the complexity is reduced to $\mathcal{O}(\log n)$. These tree based search algorithms are not designed for use in motion planning, but for database query algorithms, for which you may not know the structure of the data or the size of $n$. They are made for fast queries and do not consider the time it takes to pre-process the database to obtain the fast query. This can be a problem for motion planning algorithms, which need to do the pre-processing on-line as well.

A desirable characteristic for on-line motion planning problems is to add as many vertices in as short a time period as possible. If the speed of the range query decreases as the tree grows larger it effectively limits how large the tree can grow within some time period, which is critical in an on-line system.

We present a simple practical algorithm for minimising the nearest neighbour search time. The algorithm is demonstrated for typical RRT motion planning problems for a mobile robot. The algorithm is based on a grid in $d$ dimensions, which becomes $d$-dimensional boxes, where each vertex belongs to a specific box. Only the relevant boxes need to be searched to find the nearest vertex. The background for this algorithm has origin in computational geometry, but has to the best of our knowledge not been exploited for motion planning. The algorithm is applicable to on-line as well as off-line applications and the benefit of this box method increases as number of vertices, that need to be added to the tree, grows.

First the algorithm is presented, and then an analysis of the time complexity versus other methods is given. Then the algorithm is demonstrated in simulations.

## 2 Methods

The structure of a very basic RRT algorithm can be seen in Algorithm 1.

**Algorithm 6** Standard RRT (see [14])

```
RRTmain()
1: Tree = q.start
2: q.new = q.start
3: while Distance(q.new , q.goal) < ErrTolerance do
4:     q.target = SampleTarget()
5:     q.nearest = NearestVertex(Tree , q.target)
6:     q.new = ExtendTowards(q.nearest,q.target)
7:     Tree.add(q.new)
8: end while
9: return Trajectory(Tree,q.new)
SampleTarget()
1: if Rand() < GoalSamplingProb then
2:     return q.goal
3: else
4:     return RandomConfiguration()
5: end if
```

The objective of the algorithm is to start from an initial configuration and find a path to a goal configuration. This is done by continuously adding vertices to a tree, which is grown from the starting configuration. To extend a tree a random point is sampled from the configuration space. Then the distances to all existing vertices are calculated, and the nearest vertex is chosen. Finally the tree is extended from the chosen vertex towards the sampled configuration. When a leaf vertex reaches within some distance of the goal location, the algorithm is stopped. When the tree becomes large, a significant part of the computation time is spend on the nearest neighbour search (the red line in Algorithm 1).

## 2.1 Minimising computation time for finding nearest vertex

The basic idea of the proposed approach, for minimising the computational complexity of finding the nearest vertex, is simple. It consist of partitioning the $d$-dimensional configuration space in a number of $d$-dimensional boxes, and only calculate the distance to other vertices in relevant boxes instead of all vertices in the whole configuration space. For simplicity, we use $k$ equally sized boxes in each dimension. That means we will get $k^d$ $d$-dimensional boxes. The algorithm is started by searching all vertices in the same box as the newly sampled vertex. Then boxes adjacent to the first box are searched and step by step boxes further and further away are searched. The algorithm terminates, when the nearest found vertex is closer than the boundary between the searched and unsearched boxes. In that case it is guaranteed that no vertex in any of the outer boxes, which have not been searched yet, can be closer than the current nearest vertex.

The algorithm is illustrated for two dimensions in Fig. E.1. Here an RRT with 100 vertices have already been grown. The first step is to sample a new vertex, which is the red dot. The corresponding box, to which it belongs, is the red hatched area. A zoomed version of this area is shown in Fig. E.2.

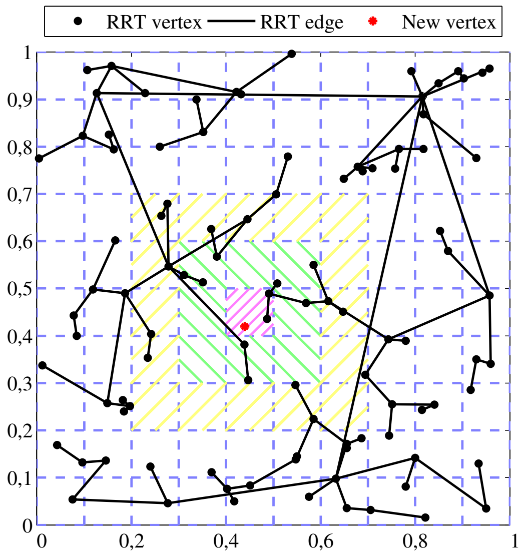


Figure E.1: The configuration space partitioned in a number of boxes in each dimension. A grown tree has to be extended with a newly sampled vertex (the red dot). The nearest boxes to the new vertex is hatched in different colours.

After finding the corresponding box, we start by calculating the distance to all vertices in this box ($V_1$ and $V_2$) and the distance to the nearest border of the box, $d_{border}$, which is shown in Fig. E.2. If any of the vertices are closer than $d_{border}$:

$$\min_{i \in I} \{|V_{new} - V_i|\} \leq d_{border} \quad , \tag{E.1}$$

where $I$ is the set of vertices in the box, then no other vertex in the configuration space can be closer. In this case the algorithm is terminated after only calculating the distance to two vertices plus the distance to the border of the box instead of calculating the distance to all 100 vertices. However, in the case of Fig. E.2 $d_{border}$ is the smallest, and thus, there may be other vertices which are closer. Therefore the distances to all vertices in the adjacent boxes are calculated, which is the green hatched area. In this case there are eight additional vertices (see Fig. E.1), for which $V_3$ in Fig. E.2 is the nearest. The distance to $V_3$ is less than the distance to the border of the yellow hatched area in Fig. E.1, which means that no other vertex can be closer, and thus $V_3$ is the nearest vertex.

In this case the algorithm terminated after calculating the distance to 10 vertices, which is $^1/_{10}$ of the total number of vertices. When initially building the tree, there are

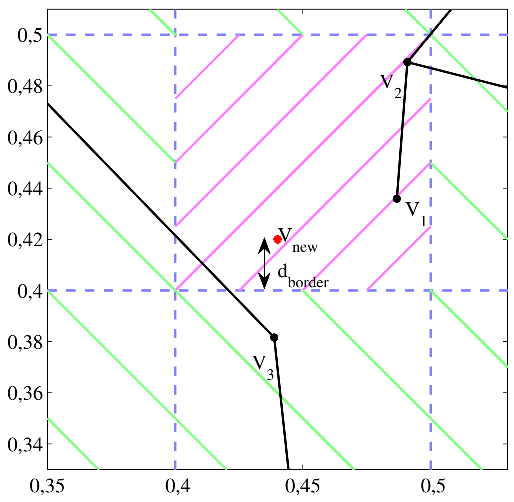


Figure E.2: A zoomed version of Fig. E.1, where the three nearest vertices and the newly sampled vertex can be seen.

only few vertices and hence a large overhead, as a large number of boxes need to be searched. But once there is a significant density of vertices, the box method is substantially faster than the brute force approach. The method is illustrated here in two dimensions, but is easily generalised to more dimensions.

The algorithm for the described box method is given in Algorithm 2.

**Algorithm 7** The algorithm for the box method, which is used to find the nearest vertex (the red line in Algorithm 1).

```
NearestVertex(Tree, q.target)
1: boxNumber = FindBoxNumber(q.target)
2: searchArea = boxNumber
3: while Dist(q.target,searchBoundary)
   < Dist(q.target,q.nearest) do
4:     ExpandSearchArea()
5:     q.nearest = FindNearest(searchArea,q.target)
6: end while
7: InsertIntoBox(q.target,boxNumber)
8: return q.nearest
```

## 2.2 Time Complexity Analysis

We start by analysing the time complexity of the standard RRT algorithm as shown in Algorithm 1. The time it takes to add $N$ vertices to the tree (disregarding the initialisation), can be calculated as the sum of the time for $N$ iterations of each of the lines 4-7 in Algorithm 1:

$$T(N) = T_{sample}(N) + T_{nearest}(N) + T_{extend}(N) + T_{add}(N), \tag{E.2}$$

where each $T_{sample}, T_{nearest}, T_{extend}, T_{add}$ correspond to the lines 4-7 respectively. $T_{sample}$ and $T_{add}$ are simple operations, and can be done in linear time, so the complexity is $\mathcal{O}(N)$. The extension time, $T_{extend}$, of the tree can take considerable longer if e.g. collision checking or other intelligent extension strategies are used. These calculations do, however, not depend on the number of vertices already in the tree, and can thus still be done in linear time. Each time the nearest vertex has to be found, the distance to all previously added vertices must be calculated:

$$T_{nearest}(N) = \sum_{i=1}^{N}(i-1)T_{dist} = \frac{N^2 - N}{2}T_{dist} \tag{E.3}$$

where $T_{dist}$ is the time it takes to calculate the distance to any other vertex. So even though $T_{dist}$ is small compared to $T_{extend}$, the complexity is $\mathcal{O}(N^2 - N)$. By adding the derived complexity for the *sample, nearest, extend* and *add* operations respectively, the combined complexity for the brute force nearest neighbour search is:

$$\mathcal{O}(N) + \mathcal{O}(N^2 - N) + \mathcal{O}(N) + \mathcal{O}(N) \approx \mathcal{O}(N^2) \tag{E.4}$$

It can be seen that the complexity is bounded by $\mathcal{O}(N^2 - N)$, which is the complexity for the *nearest* operation. So when the tree becomes large, a significant part of the computation time is spend on calculating distances to other vertices, and much computing time can be saved if the nearest neighbour search is optimised.

In the following the complexity for finding the nearest vertex, when adding vertex number $n$ is analysed. When using brute force, all $n - 1$ previous vertices need to be searched, which therefore has complexity $\mathcal{O}(n)$.

Finding the nearest vertex using the box method depends of the number of boxes $M$. If $M$ is large compared to $n$, the probability of having to search many of boxes is large, so this is not a good solution. Opposite, if $M \leq n$ the probability of having to search many boxes is small. In the case of evenly distributed vertices the average number of vertices to check will be $n/M$, and thus the complexity will be $\mathcal{O}(n/M)$. So optimally, $M$ needs to be chosen approximately equal to $n$ [11], which gives a complexity of $\mathcal{O}(1)$. However, as $n$ increases in each iteration, this is not possible all the time. But experimentally we have found that a good value for $M$ is around $N/2$, where $N$ is the maximum number of vertices in the RRT.

While $n$ is small, there is no benefit of using the algorithm, since too many boxes have to be searched. So if only inserting into boxes until $n$ is larger than some value, increases the speed of the algorithm. In contrast to the brute force approach, there is an overhead when each vertex has to be inserted into the data structure, which contains the boxes. Calculating the corresponding box for a vertex in the box method takes constant time, so the insertion has complexity $\mathcal{O}(1)$, which does not change the overall complexity ($\mathcal{O}(1)$) of the method.

This complexity is compared to the most used tree based approach for minimising the time it takes to find the nearest neighbour in motion planning problems, namely the $kd$-tree. According to [12], a $kd$-tree has a complexity for inserting a vertex of $\mathcal{O}(\log n)$. Furthermore, the complexity for finding the nearest vertex is $\mathcal{O}(n^{1-\frac{1}{d}})$, where $d$ is the number of dimensions [12]. The combined complexity for searching and inserting is therefore $\mathcal{O}(n^{1-\frac{1}{d}} + \log n) \approx \mathcal{O}(n^{1-\frac{1}{d}})$.

This complexity analysis is based on a balanced tree. A disadvantage of the $kd$-tree is that if initial vertices are not well distributed, the tree will become unbalanced, and thus the performance degrades significantly. This can be somewhat compensated for if the tree is rebuilt after some iterations. The rebuild process has a complexity of $\mathcal{O}(n \log n)$, which does not change the amortised time complexity, since it is only done once or a couple of times. Using the box method, this not a problem, since each vertex belongs to one specific box, and the data structure is therefore always the same, and nothing is gained by rebuilding.

So for finding the nearest other vertex the complexity for the brute force method is $\mathcal{O}(n)$, the $kd$-tree has a complexity of $\mathcal{O}(n^{1-\frac{1}{d}})$, and if a proper number of boxes are chosen the box method has a complexity of $\mathcal{O}(1)$.

## 2.3 General Considerations

One disadvantage with both the box based algorithm and the $kd$-tree, is the overhead when only adding a few vertices. However, this happens only in limited cases, where the configuration space is relatively small and with few obstacles, in which case an RRT may not be the right algorithm to use anyway. So to obtain good exploration in general, it is useful to use as many vertices as possible, for which the box method performs well. This is especially the case in on-line applications, where there is a time limit for how long the planning can take. It is though, possible to overcome some of the initial overhead using the box based algorithm. Initially when the number of vertices is much smaller than the number of boxes ($n \ll M$), it is faster to just add vertices to the boxes, and do a brute force nearest neighbour search. When $n$ grows, the algorithm can switch to searching in the boxes. Since the time it takes to add a vertex is small and constant (the complexity is $\mathcal{O}(1)$), it does not cost much overhead. It is also a possibility to dynamically increase the number of boxes in each dimension, as the tree grows large. This will enable the ratio ${}^{n}/_{M}$ to stay close to an optimal value.

Another advantage of the box based algorithm is that it is easy to insert or remove vertices from the data structure compared to inserting and removing from the $kd$-tree. Inserting or removing vertices only requires the operation to add or remove the vertex from the corresponding box, which is $\mathcal{O}(1)$. Inserting a vertex in the $kd$-tree requires an $\mathcal{O}(log(n))$ search to find out where to insert. Removal of vertices is generally not easy for the $kd$-tree. It requires rebuilding the whole subtree beneath the place, where the vertex needs to be inserted. Conceptually it is also easier to understand and implement the nearest neighbour search using the box method, because a $kd$-tree search includes recursive visiting branches of the tree.

## 3 Experiments

Two different experiments have been set up to demonstrate the effectiveness of the box based nearest neighbour search algorithm in comparison to the brute force approach. First the box based algorithm is implemented together with a basic RRT algorithm, as shown in Algorithm 1, and the performance is compared to the brute force approach and the $kd$-tree. Secondly, the algorithm is used in a more realistic environment, where a robot plans a path in an environment with obstacles.

In the first experiment an RRT with 10000 vertices is build in a $d$-dimensional space of $1m$ in each dimension. This is done for $d = 2, 4, 6$ dimensions using a grid with 10 boxes in each dimension. The time for adding each vertex is continuously logged, and for comparison the same experiment is done using a brute force approach. For comparison to the $kd$-tree, an existing MATLAB implementation of the algorithm has been used [15]. However, this implementation of the algorithm does not support adding vertices to an existing tree, and the nearest neighbour search relies on balanced tree. To be able to compare to the other algorithms, a new tree is built each time a vertex is added, but only the time it takes to search for the nearest vertex, is then logged for the purpose of the experiment. This is only done for the two dimensional case.

In the second experiment a robot trajectory is planned through a two dimensional configuration space. The configuration space contains obstacles represented by a potential field, and vertices are pruned when the cost gets too high. The potential field represents persons moving in the area. For further information on the setup, see [16]. Fig. E.3 illustrates an example of the potential field with a few vertices already added to the tree. The green dot to the left at $(2, 0)$ is the starting position of the robot, the red lines are possible trajectories, and the red dots are vertices. Furthermore, the red areas of the potential field are where there are persons, and hence where the robot should not go. First the tree with 10000 vertices is built using the box based distance search method. Then a tree is built using a brute force nearest neighbour search approach, where the algorithm is allowed to use the same amount of time it took to build the first tree. This experiment makes it possible to see the benefits of the box based method in an on-line application.

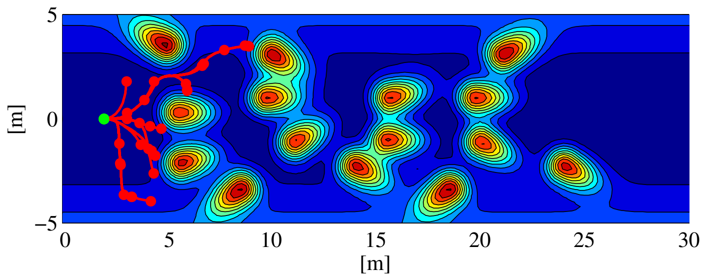


Figure E.3: An example of an RRT with 300 vertices in a potential field landscape, which is used for experimenting with the algorithm.

## 4 Results

The time for building an RRT with 10000 vertices in two, four and six dimensions is shown in Fig. E.4 using both the box algorithm and the brute force approach.

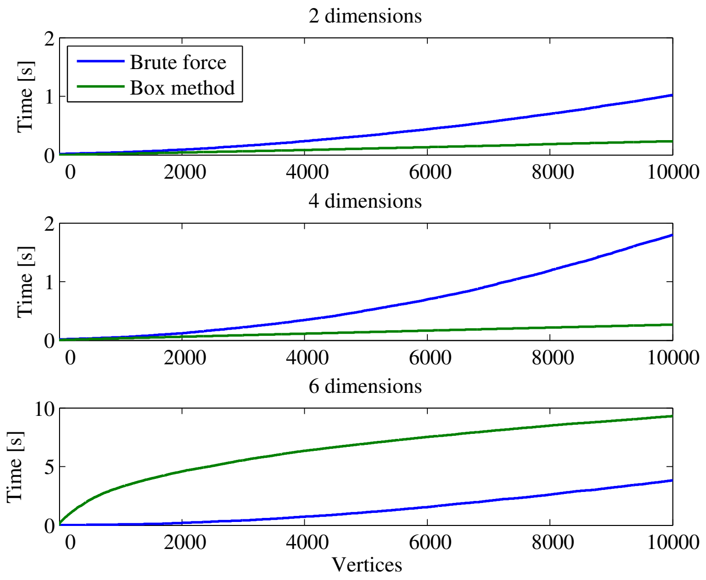


Figure E.4: From top to bottom the figure shows the time it takes for adding 10000 vertices in two, four and six dimensions respectively. The blue line shows the time it takes for the brute force approach and the green line shows the time for the box based method.

A comparison of the nearest vertex search times only, is done for two dimensions in Fig. E.5 for the three methods; brute force, box based method, and $kd$-tree.

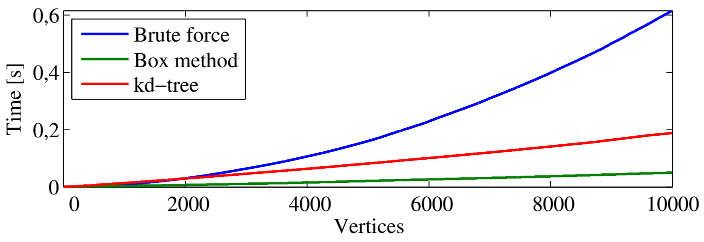


Figure E.5: Comparison of the time it takes to find the nearest neighbour for two dimensions using the $kd$-tree as well as the brute force method and the box method. Note that comparing to Fig. E.4, the brute force and the box method are slightly faster, because only the time for finding the nearest vertex and not the other parts of the RRT algorithm is considered in this experiment.

In Fig. E.6 an RRT with 10000 vertices is built in the potential field shown in Fig. E.3. And in Fig. E.7 the RRT is built using the brute force method for the same amount of time as used in Fig. E.6 for the box method. In both figures only $1/10$ of the vertices are plotted to avoid cluttering the figure too much.

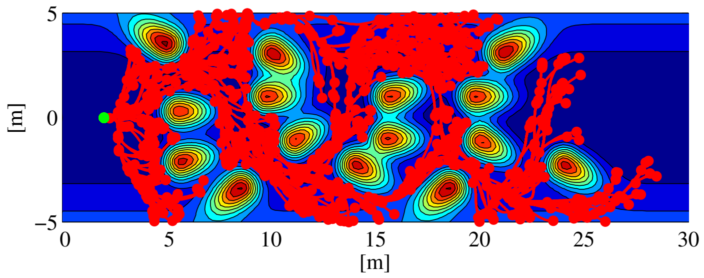


Figure E.6: A tree with 10000 vertices. Here the configuration space is explored very well.

## 5 Discussion

The results demonstrate that the box algorithm clearly outperforms the brute force approach for two and four dimensions (the top and middle plot in Fig. E.4). One thing to notice is that the box method has a larger advantage in four dimensions, than in two dimensions. Since there are 10 boxes in each dimension, there are 100 boxes in two dimensions and $10^4$ boxes in four dimensions. As the optimal number of boxes is around

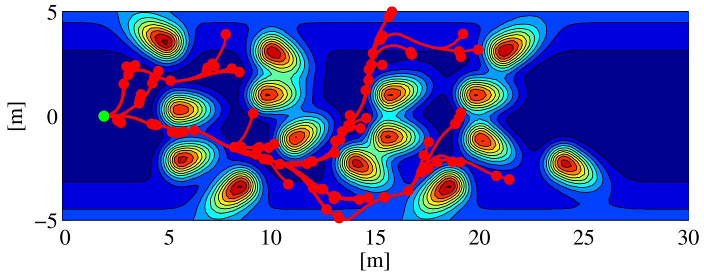


Figure E.7: A tree with approximately 2000 vertices, which does not explore the configuration space very well. This tree is build using brute force search, and has used the same runtime as the box based method in Fig. E.6

$^{N}/_{2}$, the number of boxes in four dimensions is closer to the optimal number of boxes, which is approximately 5000. In the bottom plot of Fig. E.4 it is seen that in 6 dimensions, it starts to take time for the box algorithm. This is a consequence of there being too many boxes compared to the number of vertices. The total number of boxes is $10^6$. So if the number of vertices is in the range of a million, the algorithm would perform much better, which can also be seen on the trend of the figure. After for example 30000 vertices, which is not shown in Fig. E.4, it is about twice a fast as the brute force method for six dimensions and at 300000 vertices it is 22 times faster. A way to reduce the computation time for this large number of boxes is to only subdivide some dimensions into boxes, like for example only the first three dimensions. This can also be utilised in planning problems, where the variance of the vertex locations varies much in different dimensions. It can be an advantage to have a larger number of boxes in the dimensions, were there is a large vertex variance. Possible examples are for a multilinked arm, where the first joint probably moves close around a nominal operating point, or for a mobile robot, where it may be desirable to explore the position part of the configuration space well, but e.g. the speed and orientation are around nominal values, or for configuration spaces with narrow passages in some dimensions.

According to the complexity calculations, the box based algorithm should have an approximate linear performance ($\mathcal{O}(N)$) when adding $N$ vertices, if the correct number of boxes is chosen. In the middle figure of Fig. E.4 it can be seen that the box method performs almost linearly, which is in accordance with the theory. Similarly in the bottom figure for six dimensions, it is seen that the complexity is starting to get closer to linear as the number of vertices grows larger, even though there is a large overhead in the beginning.

The performance of the $kd$-tree implementation in Fig. E.5 is seen to be worse than using the box based method. The used implementation of the $kd$-tree ensures that the search is always done in a balanced tree, which would not typically be the case in a real application, and thus in a real application, the $kd$-tree would perform worse than in this case. Furthermore, because of the complexity of the $kd$-tree search algorithm, $\mathcal{O}(n^{1-\frac{1}{d}})$, it performs worse in higher dimensions, where it can be seen in Fig. E.4 that the box based method performs better for e.g. four dimensions. The speed of the $kd$-tree algorithm can, however, be improved for high dimensional search spaces by using approximate nearest neighbour (ANN) algorithms. This comes at a tradeoff for not being entirely sure that it is the correct nearest neighbour, which has been found, which is not desirable in some applications. Additionally the bound on the complexity for ANN is $\mathcal{O}(\log n)$ [8, 17], which is sill not better than the proposed method.

In Fig. E.6 a tree with $10000$ vertices is built. It is seen that the tree covers all of the obstacle free configuration space well. Contrary to this, Fig. E.7 shows that the configuration space is not covered well, when planning using the brute force method for the same amount of time. Since using the box method cause much denser population of vertices, it is possible to choose trajectories, which are better. For example, it is seen in Fig. E.7 that the tree tends to move along narrow branches in the middle between obstacles, and might thus not find narrow passages or a feasible trajectory closer to obstacles. It is hence clearly advantageous to use the box method to be able to explore the configuration space well.

In general we argue that the presented box based nearest neighbour search algorithm is better than both a brute force approach and the $kd$-tree for robot trajectory planning. The algorithm can be adapted to perform optimal for a given problem, e.g. the number of boxes in each dimension can be adjusted to improve performance, which is not possible for the $kd$-tree. It is also easier to implement than the $kd$-tree, and it is faster to insert and remove vertices from the data structure.

## 6 Conclusion

In this paper we presented a practical algorithm for minimising the computational complexity of an RRT algorithm for robot path planning problems. It is shown that the time it takes to find the nearest neighbour in a standard RRT, using a brute force approach is substantial when the number of vertices in the tree grows. Thus, a lot of computation time can potentially be saved, if the nearest neighbour search is minimised. The proposed algorithm partitions the configuration space into a number of boxes, where only relevant boxes need to be searched to find the nearest vertex. The algorithm can be tuned to work better if an approximate bound on the maximum number of vertices is known. But generally the algorithm works better the larger the number of vertices there are in the tree. This also means that there is a relatively large overhead when only a small number of vertices needs to be added to the tree, which is not often the case. However, complexity calculations and simulations show that the proposed box based method performs better than both a brute force approach and using a $kd$-tree to find the nearest neighbour.

The algorithm can be used to increase the number of vertices, which it is possible to ad within a given time period, and can thus provide better trajectories, or compute a trajectory faster.

## References

[1] S. LaValle, *Planning algorithms.* Cambridge Univ Pr, 2006.

[2] S. LaValle and J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, p. 378, 2001.

[3] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, "Motion planning for urban driving using rrt," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, 2008, pp. 1681–1686.

[4] D. Ferguson and A. Stentz, "Anytime, dynamic planning in high-dimensional search spaces," in *Proc. IEEE International Conference on Robotics and Automation*, 2007, pp. 1310–1315.

[5] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," *Robotics Research*, vol. 15, pp. 365–374, 2005.

[6] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 2859–2865.

[7] R. Tedrake, "Lqr-trees: Feedback motion planning on sparse randomized trees," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[8] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[9] S. Arya and M. David, "Mount, Approximate range searching," *Computational Geometry: Theory and Applications*, vol. 17, no. 3-4, pp. 135–152, 2000.

[10] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[11] R. Sedgewick, *Algorithms in C*, M. A. Harrison, Ed. Addison-Wesley, Reading, MA, 1990.

[12] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press, June 2005. [Online]. Available: http://www.worldcat.org/isbn/0262033275

[13] S. Arya and D. Mount, “Approximate nearest neighbor queries in fixed dimensions,” in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1993, p. 280.

[14] D. Ferguson and A. Stentz, “Anytime rrts,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[15] A. Tagliasacchi, “kd-tree for matlab,” MATLAB Central File Exchange, Retrieved July 2010. [Online]. Available: http://www.mathworks.se/matlabcentral/fileexchange/21512

[16] M. Svenstrup, T. Bak, and H. J. Andersen, “Trajectory planning for robots in dynamic human environments,” in *IROS 2010: The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010.

[17] A. Yershova and S. M. LaValle, “Improving motion-planning algorithms by efficient nearest-neighbor searching,” *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 151 –157, feb. 2007.

# Appendix I

**Definition of Terms**

This appendix describes some of the terms, which are used throughout the thesis to avoid ambiguity in how people interpret the terms. The terms are defined according to their use in the thesis

## 1 Robotics and Planning

Terms related to planning are loosely based on the definitions in [1], [2] and [3].

**Configuration space** is the set of all possible positions a physical system (a robot) can attain. The configuration vector, $\boldsymbol{q}$, is the minimal set of parameters that specify any unique configuration of the robot. The configuration space $\mathcal{C}$ is the set of all possible $\boldsymbol{q}$. A typical example is a planar two joint arm, where the configuration is defined by the coordinates of the two joint angles $\boldsymbol{q} = [\theta_1\ \theta_2]^T$. Another example is a mobile robot, which has the position coordinates and orientation as configuration coordinates $\boldsymbol{q} = [p_x\ p_y\ \theta]^T$. The configuration space is typically used for path planning, since finding an obstacle free path for a robot in a physical world, reduces to finding a path of a point through the obstacle free configuration space $\mathcal{C}_{free}$.

The configuration space is not to be confused with the *state space* $\boldsymbol{x}$. While the configuration parameters are uniquely defined, the state space can be any set of chosen parameters that describe the state of the robot. For example a typical state, in addition to the position and orientation, also contains parameters for velocity.

**Non-holonomic** was originally introduced to describe the problem of motion planning for wheeled robots. Non-holonomic refers to constraints on the configuration space that can not be completely integrated, which means that the constraints involves derivatives of the configuration $\boldsymbol{q}$. This means that constraints written as $\Phi(\boldsymbol{q}) = 0$ are holonomic constraints and constraints written as $\Phi(\boldsymbol{q}, \dot{\boldsymbol{q}}) = 0$ are non-holonomic constrains. For example a non-holonomic constraint for a car like robot is the constraint that it cannot move sideways. If $\boldsymbol{q} = [p_x\ p_y\ \theta]^T$, this can be written as

$$\dot{\boldsymbol{q}}^T \begin{bmatrix} \sin q_3 \\ \cos q_3 \\ 0 \end{bmatrix} = 0 \quad . \tag{I.1}$$

A holonomic constraint is for example limitations on the joint angles of a robotic arm.

Non-holonomic systems are closely related to underactuated systems, and the terms are often used interchangeably. This is however not a correct usage even though an underactuated system is often non-holonomic and vice versa. The definition of underactuated is that the number of actuators is strictly less than the dimension of $\mathcal{C}$. For example a unicycle type robot has a 3-dimensional configuration space, but can only control rotation and speed, and is thus underactuated. It is also non-holonomic because it cannot drive sideways. However, if the unicycle were transformed into a hover craft, it would be able to slide sideways. Therefore the constraint on the sideways motion would be removed, and it would be holonomic. However, it would still be underactuated, and thus underactuated does not imply non-holonomic.

**Kinodynamic planning** originally comes from [4] and refers to planning that takes into account both the kinematics and the dynamics of the robot, i.e. the velocity and acceleration bounds must be satisfied. This typically involves second order constraints on $\mathcal{C}$, e.g. the input controls the acceleration $\ddot{\boldsymbol{q}} = u$. Kinodynamic planning is not necessarily encompassed by non-holonomic planning, but it is often the case when working with underactuated systems.

**Motion planning** is related to problems involving the local motion of the robot. It can be avoiding local obstacles or adapting the motion according to humans close to the robot.

**Trajectory planning** has a longer time horizon than motion planning. In trajectory planning the goal is to create a trajectory through $\mathcal{C}_{free}$, the obstacle free space. A trajectory planner is an algorithm that gives both a path and a velocity that should be executed.

**Trajectory generation** is analogous to trajectory planning. A trajectory planning algorithm generates a trajectory to be executed.

**Path planning** it the process of finding a path through $\mathcal{C}_{free}$. A path disregards kinematic constraints and dynamics and only find a way to get the robot from A to B.

**Navigation** is the process of executing a planned trajectory. The robot navigates through the environment while continuously replanning the trajectory.

**Pose** sometimes have an ambiguous meaning. Sometimes it means only the orientation, and sometimes a 3-d orientation (yaw, pitch, roll) of e.g. a face, and sometimes the configuration of the whole body of a person. In this thesis pose refers to the position an orientation of a person or robot $[p_x \; p_y \; \theta]^T$. The orientation of a person is defined to be the direction of the torso, and not the direction of the face. This is because the orientation of the face can change frequently without it actually have any meaning in relation to where the person is going. The orientation is also not exactly the same as the motion direction. This can be exemplified by a person moving slowly backwards. In this case, the orientation is in the opposite direction of the motion.

**Tree Nodes and Vertices** are two words, which are often used interchangeably. However, in this thesis there is a slight distinction. A tree consist of a number of nodes and edges connecting the nodes. So a tree is completely specified by which nodes are connected to which nodes. So when drawing the tree with nodes, it has no meaning how they are positioned, but only how they are connected. On the other hand, vertices can contain a lot more information. This can for example be velocity and associated trajectory cost. Tuples illustrating this are shown in Eq. (I.3).

$$\begin{aligned} node &= \{parent, children\} \qquad (I.2) \\ vertex &= \{parent, children, position, orientation, velocity, cost \ldots\} \end{aligned}$$

## 2 Human-Robot Interaction

**Open-ended environment** defines an environment, where it is not completely defined what can happen. A human environment is unstructured and there are no restrictions to what may happen around the corner. Thus an open-ended environment is where all possible scenarios cannot be modelled.

**Close interaction** is used in the thesis as interaction that does not involve moving around in the environment. That is for example speech, manipulation tasks, gesture recognition, or facial expressions.

**Interaction session** denotes the process from a robot detects a person and until the person is gone or close interaction is initiated. That means that it is the time period, where the robot and the person mutually adapts their motion to each other.

**Interest in interaction** is defined as a person's interest in interaction, as perceived by the robot. It should therefore not be interpreted as the true internal state of the person, which would require more in depth psychological research.

**Person Indication (*PI*)** is a fuzzy variable that gives an indication to whether a person is interested in close interaction with the robot. *PI* represents the robot's belief in the interest level of the person. It is a continuous variable that belongs to the interval between $0$ and $1$ (*PI* $\in [0; 1]$). *PI*$= 1$ represents the case, where the robot is certain that the person wants to interact and if *PI*$= 0$ the robot is certain that the robot does not want to interact. The closer *PI* is to $0.5$, the more uncertain the robot is to whether a person is interested in interaction.

**Person Skill Indication (*PSI*)** is a fuzzy variable that resembles *PI*. It is used when the robot plays a game with a person. *PSI* represents the skill of the playing person. *PSI* is a continuous variable that belongs to the interval between $0$ and $1$ (*PSI* $\in [0; 1]$). When *PSI*$\approx 1$ the robot thinks that it is a skilled player, who is playing, and opposite it is a bad player if *PSI*$\approx 0$.

## References

[1] S. LaValle, *Planning algorithms*. Cambridge Univ Pr, 2006.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[4] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, p. 1066, 1993.