

Imperial College of Science, Technology and Medicine
Hamlyn Centre Institute

Development of New Intelligent Autonomous Robotic Assistant for Hospitals

Alexandre Paschoal Vicente

Supervised by Professor Guang-Zhong Yang

and Dr Jindong Liu

A thesis submitted in partial fulfilment of the requirements for the degree of
PhD in Medical Robotics and for the Diploma of Imperial college, August 2018

Imperial College, London

Abstract

Continuous technological development in modern societies has increased the quality of life and average life-span of people. This imposes an extra burden on the current healthcare infrastructure, which also creates the opportunity for developing new, autonomous, assistive robots to help alleviate this extra workload.

The research question explored the extent to which a prototypical robotic platform can be created and how it may be implemented in a hospital environment with the aim to assist the hospital staff with daily tasks, such as guiding patients and visitors, following patients to ensure safety, and making deliveries to and from rooms and workstations.

In terms of major contributions, this thesis outlines five domains of the development of an actual robotic assistant prototype. Firstly, a comprehensive schematic design is presented in which mechanical, electrical, motor control and kinematics solutions have been examined in detail. Next, a new method has been proposed for assessing the intrinsic properties of different flooring-types using machine learning to classify mechanical vibrations. Thirdly, the technical challenge of enabling the robot to simultaneously map and localise itself in a dynamic environment has been addressed, whereby leg detection is introduced to ensure that, whilst mapping, the robot is able to distinguish between people and the background. The fourth contribution is geometric collision prediction into stabilised dynamic navigation methods, thus optimising the navigation ability to update real-time path planning in a dynamic environment. Lastly, the problem of detecting gaze at long distances has been addressed by means of a new eye-tracking hardware solution which combines infra-red eye tracking and depth sensing.

The research serves both to provide a template for the development of comprehensive mobile assistive-robot solutions, and to address some of the inherent challenges currently present in introducing autonomous assistive robots in hospital environments.

Declaration of Originality

I hereby declare that the research presented in this thesis is my own, unless where appropriately referenced.

Alexandre Paschoal Vicente

©The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work

Acknowledgements

First, I would like to thank Professor Guang-Zhong Yang, whose mentoring and support allowed me to finish this Ph.D. I would like to thank all the people who helped me throughout this long journey, and to extend an extra thank you to those who really played an important part in my moments of need.

I would like to thank my family, especially my mother, for your unconditional love, your support, and your words of encouragement. To Jindong, for always being there for me, for believing in me, and guiding me when I needed it the most. To Max Eames, for helping me to focus on my goals and providing words of wisdom in moments of great sadness and difficulty.

Last, but certainly not least, I would like to thank my wife Yi-Chen, from the bottom of my heart, for being my brightest light throughout my darkest times. Even when we were apart, I always felt you by my side Xiao Mao Mi, always.

Contents

Abstract.....	3
Declaration of Originality	5
Acknowledgements	7
List of Acronyms	14
List of Figures.....	15
1 Introduction.....	22
1.1 Assistive Robotics in Healthcare	23
1.2 Structure of the Thesis	25
1.3 Research Contributions	27
2 Background	29
2.1 Introduction.....	29
2.2 Mapping and Localisation in Mobile Robotics.....	33
2.3 People detection	36
2.4 Navigation.....	39
2.4.1 Global Planning.....	39
2.4.2 Local Planning	42
2.5 Environment Perception.....	45
2.6 Human-Robot-Interaction (HRI).....	47
2.6.1 Interaction roles.....	47
2.6.2 Robot safety	48
2.6.3 Gaze Detection.....	49

2.6.4	Intention recognition	52
2.7	Conclusions.....	54
3	Mobile Robot Hardware Design	57
3.1	Introduction.....	57
3.2	Mechanical Design.....	58
3.3	Base Kinematics.....	60
3.3.1	Forward and Inverse Kinematics	61
3.3.2	Inverse Kinematics.....	64
3.3.3	Base Velocity Close Loop Control	64
3.3.4	Prototypes and Suspension Test.....	65
3.4	CAD Designs of the Chassis.....	66
3.5	Suspension test.....	69
3.6	Electrical and Electronic Design.....	70
3.6.1	Power Grid system.....	70
3.6.2	Batteries	72
3.6.3	Sensors	74
3.7	Conclusions.....	75
4	Floor classification	77
4.1	Introduction.....	77
4.2	Data collection	79
4.3	Feature Extraction.....	81
4.3.1	Statistical Features	82
4.3.2	Frequency-based Features.....	82

4.3.3	Final PCA Feature Set.....	83
4.4	Training and Classification	85
4.5	Results.....	86
4.5.1	Classifier Performance	86
4.5.2	Real Scenario Validation	88
4.6	Conclusions.....	89
5	Dynamic Mapping.....	90
5.1	Introduction.....	90
5.2	HectorSLAM.....	91
5.3	Leg Detection.....	94
5.3.1	Segmentation and Clustering	94
5.3.2	Cluster Classification	98
5.4	Results.....	100
5.4.1	Detecting people	100
5.4.2	Improvement on Mapping by filtering persons in a real-world scenario.....	100
5.5	Conclusion	103
6	Dynamic Navigation.....	104
6.1	Introduction.....	104
6.2	Position Initialisation State	105
6.3	Continuous Localisation Using Scan Alignment	106
6.4	Tracking Position and Movement of People.....	107
6.5	Path Planning Methods	108
6.5.1	Safe Interval Path Planning (SIPP)	110

6.6	Anticipated Improvement Over Previous Methods: Dynamic Safe Interval Path Planning (DSIPP)	115
6.6.1	Trajectory Update	115
6.6.2	Geometric Collision Detection.....	118
6.7	Experiments and Results.....	121
6.8	Conclusions.....	124
7	Sensor Fusion for Long Range Gaze Estimation	127
7.1	Introduction.....	127
7.2	Gaze Detection System Design.....	129
7.2.1	Control Board Design	132
7.2.2	Prototypes	133
7.3	Depth-Based Gaze Estimation	135
7.3.1	Eyes Gaze Vector.....	135
7.3.2	Nose Gaze Vector	136
7.3.3	Forehead Gaze Vector.....	137
7.4	Proposed Sensor Fusion Gaze Estimation	139
7.5	Experiments and Results.....	143
7.5.1	Experiments for Depth-Based Gaze-Estimation	144
7.5.2	Sensor Fusion Results	146
7.5.3	Real-World Evaluation	148
7.6	Conclusions.....	151
8	Conclusions.....	153
8.1	Technical Contributions of Thesis	155

8.2	Future Work.....	158
	Bibliography	160

List of Acronyms

AMCL	Adaptive Monte Carlo Localisation
CAD	Computer Aided Design (3D model)
CCD	Charge-Coupled Device
CMOS	Complementary Metal Oxide Silicon (sensor)
DWA	Dynamic Window Approach
ELM	Extreme Learning Machine
GMM	Gaussian Mixture Model
GP	Gaussian Process
GPIO	General Purpose Input / Output
HCI	Human-Computer Interaction
HMM	Hidden Markov Models
HOG	Histogram of Gradients
HRI	Human-Robot Interface
ICP	Iterative Closest Point
ICSs	Inevitable Collision States
IMU	Inertial Measurement Unit
IR	Infrared
IRL	Inverse Reinforcement Learning
LDA	Linear Discriminant Analysis
MCL	Monte Carlo Localisation
ML	Machine Learning
NN	(Artificial) Neural Network
PCSs	Probability Collision States
PID	Proportional Integral Derivative (controller)
PSD	Power Spectrum Density
QDA	Quadratic Discriminant Analysis
RL	Reinforcement Learning
RFS	Random Finite Set
ROS	Robot Operating System
RRT	Rapidly-Exploring Random Tree
SAR	Socially Assistive Robot
SIPP	Safe Interval Path Planning
SLAM	Simultaneous Localisation and Mapping
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine
TOF	Time-of-Flight (camera)
USB	Universal Serial Bus
VFH	Vector Field Histogram
VOs	Velocity Obstacles

List of Figures

Figure 1.1 – List of different disabilities in UK population in 2012 [2]	22
Figure 1.2 – Examples of robotics in current healthcare. (a) Da Vinci robot and its master station, used in robotic assisted surgeries. (b) RP-7 telepresence robot used by clinicians to assess patients remotely	24
Figure 1.3 – Common roles an assistive mobile robot may take in a hospital environment including, but not limited to, guiding persons, delivering goods and checking on patients periodically	24
Figure 1.4 – Hierarchical thesis structure based on how the contents of each chapter build upon their successors.....	25
Figure 2.1 – Mobile assistive robot evolution over the years. From left to right, Minerva, Pearl, Care-o-Bot 3, RP-7, RP-Vita and Pepper.	32
Figure 2.2 – Summary of relevant people-detection techniques.....	39
Figure 2.3 – Representation of global and local planning on a map scale. The green squares represent all available free space where the robot can move. The grey squares represent obstructed space, such as a wall. The red area around the robot represents its local planning space.	39
Figure 2.4 – The Pioneer 3-AT robot platform; used in different research on outdoor surface classification	45
Figure 3.1 – Previous assistive robot prototype, codename EO3. This prototype was equipped with an off-the-shelf differential wheel base	57
Figure 3.2 - NHS Hospital Building specification based on the Department of Health. The smallest corridor should allow for a wheelchair and a person to pass, where the robot would have to fit the same space as the person (600mm).....	58
Figure 3.3 – A comparison between omni wheel and Mecanum wheel regarding the floor contact force. Blue arrows represent the direction of the floor reaction force. Red arrows represent the wheel's rotation axis (X axis). Green arrows on the 2D diagram represent the Y-axis.	61

Figure 3.4 – Normal base orientation for the omni wheel (a), and the Mecanum wheel (b) versions of the base	62
Figure 3.5 – Rotated base orientation for the omni wheels (a), and the Mecanum wheels (b) versions of the base. When the frames of reference are rotated, they align with the robot's X axis.....	63
Figure 3.6 – Base velocity close loop control with software (in white) and hardware (in grey) elements.	64
Figure 3.7 – First suspension design concept. The base can be seen from an isometric view (a), a front view (b) and a top view (c)	67
Figure 3.8 – Second suspension design concept. The base can be seen from an isometric view (a), a front view (b) and a top view (c).....	67
Figure 3.9 – Third and final design. Pre-rendered view (a) along with the top view (b) and side view (c) showing the two parts of the suspension in pink and blue.	68
Figure 3.10 – Demonstration on how the swivel suspension works. Notice that the front wheels rotate around the centre axis, all four wheels maintaining contact on the floor and obstacle.....	69
Figure 3.11 – Simplified power diagram grid. Brown and blue lines indicate LIVE and NEUTRAL line when connected to a power outlet. Red lines indicated 24V and yellow lines indicated 12V.....	71
Figure 3.12 – Pictures of the battery management system and its components in detail.....	73
Figure 3.13 – Voltage and temperature levels of individual battery cells through the BMS.....	73
Figure 3.14 – Sensor positions and sensing regions on the robot. The side view shows the depth camera's field of view and the Lidar's scanning plane parallel to the ground. The top view shows the angle coverage of both Lidars, although with their range radius reduced to fit the diagram.....	74
Figure 3.15 – Comparison table between the physical characteristics of the proposed New Assistive Robot and two of the latest assistive robots used in hospital environments	76
Figure 4.1 – IMU unit in detail and its location on the robot's chassis. The Z arrow on the IMU indicates the vertical direction of the vibration.....	78
Figure 4.2 – Complete data processing and classification pipeline. The features extracted from the sample in the shifting window are reduced and normalised before being used for training.	

Visualisations of the data in each stage are shown to illustrate the process, but their meaning is covered in detail in the subsequent sections.	79
Figure 4.3 – Example of vibration recorded by the IMU during recording stages: (A) robot not moving; (B) robot moving in one direction; and (C) robot rotating	80
Figure 4.4 – All four types of movement were executed on each floor to ensure coverage of all degrees of freedom of the base. Orange arrows indicate the front side of the robot.....	80
Figure 4.5 – The four different type of floors used for training and validation. Carpet (a), PVC (b), Wood (c) and Granite (d).....	81
Figure 4.6 – Power Spectrum Density of signal recorded while the robot performed the same movement at different speeds - shown in different colours	83
Figure 4.7 – Cumulative percentage explained by the principal components. Statistical and PSD features, in blue and green respectively, are converted into PCA components whose 10 first components can recreate 95% of the original features.....	84
Figure 4.8 – PCA feature comparison between the four types of floors. The four rectangular patches represent the first 25 PCA components of 11 sample windows for each floor. Values are visualised as colours, where zero is dark blue and one is dark red	84
Figure 4.9 – Final overall classification using all data points from all recorded speeds and movements combined. ELM shows a higher classification accuracy with only 10 data components compare to the other classifiers	87
Figure 4.10 – Confusion matrix generated using the Extreme Learning Machine classifier using 200 neurons.....	88
Figure 4.11 – Map of the robot's trajectory on the floor; green colour indicates carpet and blue indicates plastic surface	88
Figure 4.12 – Recorded floor classification over time. The detected floor (red) is compared to the ground truth (blue), where the peaks and dips on the red plot represent mis-classifications.....	89
Figure 5.1 – The raw scan is segmented in to into clusters of nearby points that are then classified as a person, and used for tracking, or as part of the background, and used for mapping.....	90

Figure 5.2 – Different resolution maps created using HectorSLAM. The larger the cell size, the lower the level of detail of the map.....	92
Figure 5.3 – Bilinear interpolation of the occupancy map [31]	92
Figure 5.4 – Step-by-step of segment clustering algorithm using a uniform-grid. Segments are separated and added to the uniform-grid before being clustered into the final clusters representing possible detected people.	96
Figure 5.5 – Speed comparison between uniform-grid clustering and Delaunay triangulation clustering in terms of speed	97
Figure 5.6 – Two people walking side-by-side are identified as one single individual if normal distance clustering is used.....	97
Figure 5.7 – Leg detection based on features from 2D clusters. Green areas A and B represent where the cluster of points represent legs are located. The blue dot represent the first point of each of the segment clusters.	98
Figure 5.8 – Recall to precision graph (ROC) for leg detection using the Adaboost classifier.....	100
Figure 5.9 – An actual room in which, whilst occupied, data was collected for verifying the possible improvements of the proposed new mapping method	101
Figure 5.10 – Side-by-side comparison between two versions of HectorSLAM: (a) Original HectorSLAM and (b) Proposed HectorSLAM with leg-detection pre-processing	102
Figure 6.1 – Assistive robot's main localisation routine. The continuous scan matching is used when the robot knows where it is, otherwise, the AMCL method is used to locate the lost robot.....	105
Figure 6.2 – Comparison between different weights for A*. The higher the coefficient epsilon, the faster the path is to compute, but the less optimal it becomes	109
Figure 6.3 – Collision diagram showing robot and moving obstacles in their respective trajectories, and key points indicating delimitations of collision interval over time	111
Figure 6.4 – Real-world representation of the robot's footprint discretised into cells (a) and same footprint, depicted over time, showing the robot moving in a straight line (b)	112
Figure 6.5 – Modified A* Algorithm with Safe Intervals in place [52].....	113

Figure 6.6 – “getSuccessors()” subroutine used in the modified A*[52].	113
Figure 6.7 – Diagram showing the hierarchical dependency of the path finding methods discussed in this chapter.	114
Figure 6.8 – Example of how a moving obstacle is tracked throughout its trajectory. The person is represented by the blue and their estimated trajectory is represented by the vector T.	116
Figure 6.9 – ‘Loop scenario’ where the person moves in and out of sight, forcing the robot to re-plan and repeat. This is fixed by adding a wait condition before re-planning.	117
Figure 6.10 – Collision detection using a line and circle intersection method. The points C1 and C2 represent respectively the beginning and end of the collision between the robot and the person, should the person maintain their trajectory	119
Figure 6.11 – Comparison between discrete time based and geometric based collision detection. The continuous time-step holds the entire duration of the occupied state, whereas the discrete method requires an iterating through each time step.	121
Figure 6.12 – One of the scenarios with the robot (green circle) avoiding 15 people moving across the map. The robot’s calculated path can be seen in blue, and the person’s trajectory vectors can be seen in magenta.	122
Figure 6.13 – Time taken for each update for each person that entered the robot’s line-of-sight. ASIPP becomes slower as the time horizon increases, whereas DSIPP becomes slower as the number of visible moving obstacle increases	123
Figure 6.14 – Performance improvement between ASIPP and proposed DSIPP methods shown in a 3D plot (a), and on a 2D plot (b). The blue line and red lines on the 2D plot represent the mean and one standard deviation of the number of moving obstacles for each time horizon.	124
Figure 7.1 – Illustration of a scenario where the robot is guiding an elderly patient using gaze tracking to enhance its response: (a) Robot detects that the patient is looking at it and maintains course, speed and distance from the person; (b) If the person diverts their gaze for some time, the robot can detect the person’s change in focus and reduces its speed to prepare to wait for the person in case they stop; (c) The robot waits for the person to look back at it before resuming guidance.	128

Figure 7.2 – 3D model of the final assembly version of the eye tracker. The IR camera was fixed on an aluminium extrusion, with the on-axis illumination source mounted inside the camera’s lens’ rim. The off-axis illumination sources are mounted on both ends of the bar, and the Kinect sensors is fixed at the bottom	130
Figure 7.3 – Major stages of detecting the pupils using the IR camera setup. Images takes using different illumination sources are subtracted to reveal the position of the pupils. Subject of the images is a colleague that volunteered for the experiment.....	131
Figure 7.4 - 3D Point cloud capture by the kinect of two subjects walking towards the robot. The pupils detected by the IR camera are projected onto the 3D pointcloud in order to get a real-world depth estimate of there the eyes are in space.	131
Figure 7.5 – Electronics circuit diagram for the control board, responsible for controlling both on- and off-axis light sources and interfacing with the camera	132
Figure 7.6 – IR light from centre and off-centre IR LEDs captured by a mobile phone camera. CMOS sensors, like the ones in mobile phones, are able to capture both visible and infrared light.	134
Figure 7.7 – Different prototypes of the eye tracker. The first prototype was built in a day as proof of concept. The second and third one used 3D-printed parts and represented several minor interactions in design.	134
Figure 7.8 – Top-view diagram of gaze estimation using the 3D position of the left and right eyes and nose. The robot reference frame can be seen on the left, and angle α corresponds to the XZ-plane angle of the person’s gaze.....	136
Figure 7.9 – Top-view diagram of gaze estimation using left and right eyes and nose. The robot reference frame can be seen on the left, and angle α corresponds to the XZ-plane angle of the person’s gaze.	137
Figure 7.10 – Top-view diagram of gaze estimation using left and right eyes and forehead. The robot reference frame can be seen on the left, and angle α corresponds to the XZ-plane angle of the person’s gaze.	138

Figure 7.11 – Diagram of how the position of the eyes in space are projected on the IR camera’s image plane	139
Figure 7.12 – Gaze angle estimation at different positions in front of the gaze detector. Depth and eye positions used to generate graphs were collected from a real volunteer.	142
Figure 7.13 – Diagram of how the experiments were conducted. Subjects would stand at distances of 1.5, 2.0 and 2.5 m from the robot and look at wall-markers to ensure ground-truth gaze direction...	143
Figure 7.14 – Results from static experiments. Overall errors for each distance indicate that the accuracy is inversely proportional to the distance.	145
Figure 7.15 – Average performance of the gaze detector when subjects moved towards the robot. The forehead vector estimation presented the lowest error margin compared to the eyes and nose vectors	145
Figure 7.16 – Results of the sensor fusion method compared to the forehead vector. The proposed sensor fusion method consistently demonstrated higher accuracy in estimating the angle of gaze....	147
Figure 7.17 – Sensory fusion results for a moving scenario compared to the forehead vector estimation method. The proposed sensor fusion method still outperforms the forehead vector estimation in a moving scenario.....	148
Figure 7.18 – Example of three subjects within 1.5 m of the robot. Gaze is detected using sensory fusion and processed through a Kalman Filter to reduce jitter.	149
Figure 7.19 – Static scenario with three people chatting near the robot. The gaze of persons A and C is not detected since both eyes must be visible to the gaze tracker	149
Figure 7.20 – Dynamic scenario with three people. Persons A and B walk by whilst engaged in conversation, and person C is walking by himself.....	150
Figure 7.21 – Dynamic scenario in which people are walking along a corridor whilst the robot remains stationary. The arrows represent people’s estimated gaze-direction up to 2.5 metres from the robot.	151

Chapter 1

1 Introduction

In the last 20 years, the development of modern societies, including the advances in technology, have significantly extended the overall longevity of the population. People around the world have increased their lifespan by 6 years, with children born after 2011 having 33% chances of reaching 100 years of age. In Europe alone, it is expected that by 2020, 25% of all population will be at least 60 years of age [1].

The ageing of the population, however, creates new challenges that current healthcare infrastructures in most western societies are not prepared to deal with. Elderly patients often develop chronic conditions over their lifetime that require extra care during their admission in the hospital. In addition to the ageing population, patients with disabilities also require extra assistance throughout their treatment. In the UK alone, it was recorded that nearly 12 million people with at least one type of disability in 2012 [2]. The types of disabilities recorded can be seen in Figure 1.1.

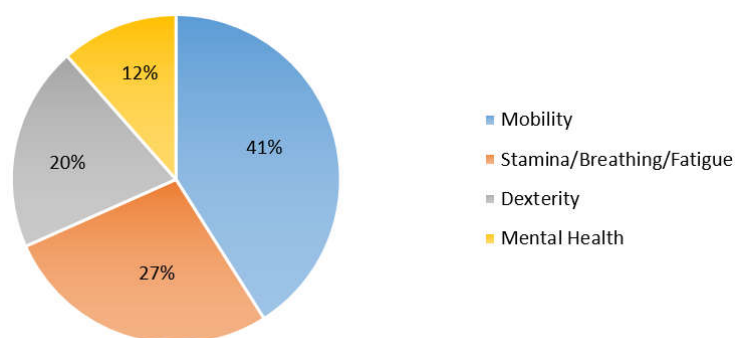


Figure 1.1 – List of different disabilities in UK population in 2012 [2]

Elderly patients and patients with some form of physical or cognitive impairment often require extra staff to help them move, eat, and manage their activities of daily living. These patients often cannot walk for themselves or require walkers and tend to perform tasks at very slow rates. They also

require more periodic checks from staff to make sure they are not at any risk to themselves. This extra care creates an excessive workload for current healthcare systems, which tends to worsen as more patients with chronic conditions are admitted. As the population ages, with the elderly expected to constitute 29.5% of the European population by 2060 [3], new solutions need to be devised to prevent declining care stands in terms of care-provision.

1.1 Assistive Robotics in Healthcare

The field of robotics has grown rapidly in the last few decades, moving past exclusive industrial applications to more general consumer roles. Robotics applications in today's healthcare mainly consist of intelligent tools designed to augment specialists' actions and perceptions, as is the case with the introduction of the *da Vinci® Surgical System*, first launched in 1999 by Intuitive Surgical® [4]. The introduction of the *da Vinci* System represented a leap in minimally invasive surgery, as it allowed surgeons to operate from a sitting position at a distance using gravity-compensated hand controllers. The *da Vinci® Surgical System* was originally designed for cardiac surgeries but today it is used in various general minimally invasive surgeries; in particular, it is highly requested for prostatectomies due to the higher level of precision it gives the surgeon while removing the target tissue [5].

Another area developed in healthcare robotics is the introduction of robots deployed to provide telepresence, such as the remotely controlled RP-7 robot, which was first unveiled in 2007 [6]. These robots are operated by medical specialists and are used to examine patients without the need for the specialist to be on-site. Such technology allows specialist knowledge to be shared at multiple sites without the need for travelling from one medical centre to another. This permits broader access to highly specialised diagnoses and permits staff with less training and expertise to perform various necessities with the confidence of the remote specialists. Such an arrangement offers time savings and cuts lead-times to a fraction of what they would be were the medical specialist required to travel to multiple sites.

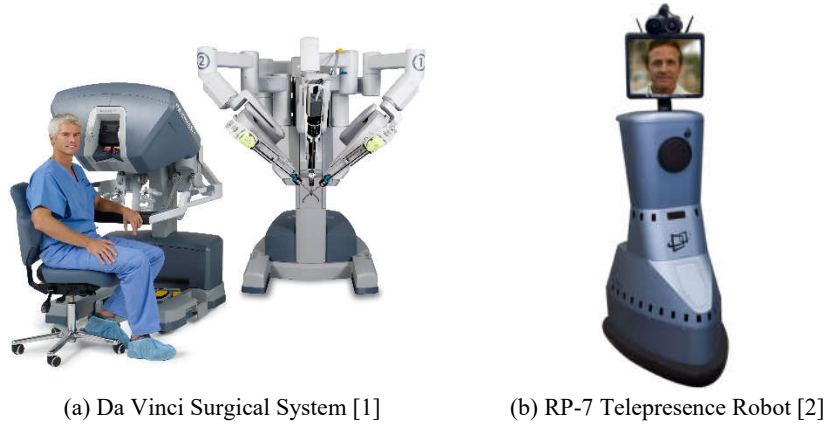


Figure 1.2 – Examples of robotics in current healthcare. (a) Da Vinci robot and its master station, used in robotic assisted surgeries. (b) RP-7 telepresence robot used by clinicians to assess patients remotely

Telepresence robots can be deployed in almost any hospital environment, but it always requires one person to operate it. The future mobile assistive robot envisioned in this research will take roles that will alleviate the workload of the current healthcare workforce in hospitals by taking on some of the tasks that require little to no expertise from staff, thus liberating nurses and other staff to focus on vital tasks such monitoring patients that require special care and attending to emergency situations.

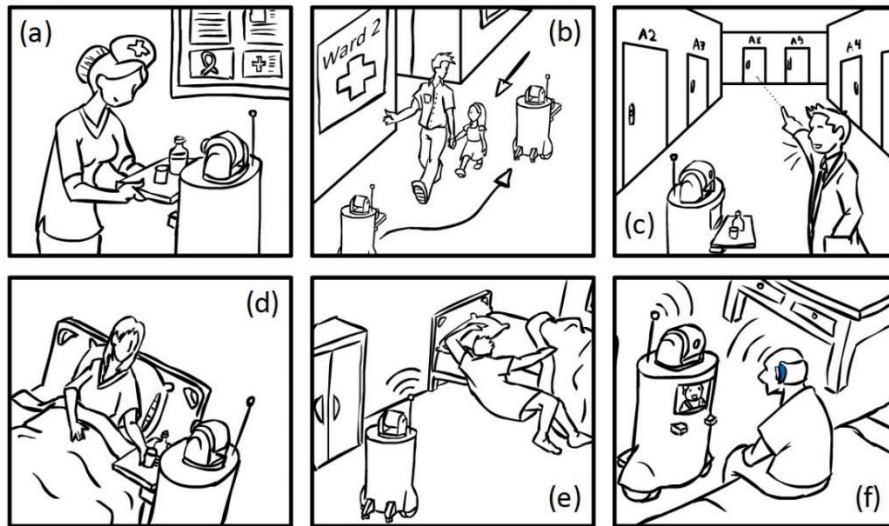


Figure 1.3 – Common roles an assistive mobile robot may take in a hospital environment including, but not limited to, guiding persons, delivering goods and checking on patients periodically

Some of the possible scenarios where the new assistive robot can be employed are depicted in Figure 1.3. The envisioned assistive robot will be able to receive ad-hoc deliveries (a) and deliver them to the patient's bed (d). The robot will also be able to move across the environment without disrupting

¹ Image from <http://www.davincisurgery.com/da-vinci-surgery/da-vinci-surgical-system/>

² Image from <https://www.roboticstoday.com/robots/rp-7>

walking visitors or staff (b) and receive voice commands with visual cues from staff in case tasks need to be changed (c). Furthermore, the assistive robot will also be able to perform autonomous periodic checks on patients to detect if they are out of their bed and/or need assistance (e). Lastly, body-sensor networks may be deployed into the standard patient monitoring procedure, thus allowing the robot to directly collect data from the patient at close range (f).

1.2 Structure of the Thesis

This thesis is organised in a hierarchical way, where each chapter lays the foundation for the next. The diagram shown in Figure 1.4 shows how the contents are linked together.

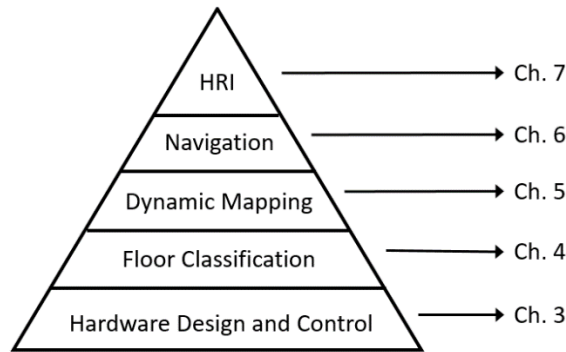


Figure 1.4 – Hierarchical thesis structure based on how the contents of each chapter build upon their successors.

The hardware design sits at the bottom as a prototype of the robot had to be built before further development could take place. Next is a summary of the contents for each of the upcoming chapters.

Chapter 2 presents a review of the current state of robotics in healthcare, along with an overview of the most relevant literature in each of the major areas relevant to the development of the assistive robot envisioned in this research.

Chapter 3 describes in detail the design stages involved in the development of the mobile assistive robot platform used in this research, from its initial concept to its final assembled form. First, the advantages of different holonomic bases are compared and verified with prototypes. Then, the overall design of the robot is presented along with the placement of the sensors required for both world mapping and people detection. Details of its electrical construction and power supply are also provided. Finally,

the kinematics and control method used on the motors are presented, followed by an overview of the final robot prototype.

Chapter 4 explains how the assistive robot platform is able to classify the type of floor underneath it while it moves around in real time. An IMU unit attached to the robot's chassis is used to collect the vibration signal while the robot moves; a method for extracting features and classifying them is presented in detail. Results demonstrate that accurate floor-type classification can be drawn from a two-second sample window. The robot is capable of differentiating between four common types of surfaces inside a hospital environment.

Chapter 5 demonstrates how a current state-of-the-art Simultaneous Localisation and Mapping (SLAM) method for static environments can be used to map dynamic environments by adding people-detection as a pre-processing stage. The proposed people-detection algorithm segments Lidar data and uses a machine learning-based classifier to identify which segments only represent the static background.

Chapter 6 introduces several improvements to dynamic navigation by building upon the localisation methods presented in Chapter 5 and improving the performance of state-of-the-art dynamic path planning. A new geometric method for detecting collision is introduced, which substitutes a previous multi-dimensional occupancy state search by a cell-to-cell collision check using only visible obstacles.

Chapter 7 shows in detail the development of a new eye tracker designed for detecting the direction of gaze in subjects more than 60 cm away from the robot. The new eye tracker combines IR eye tracking with depth sensing to achieve a more accurate estimation of the subject's position of the eyes in space, and therefore a more accurate gaze direction estimation. The hardware development, including the prototyping stage, is covered in detail. A series of experiments is also related showing the new eye-tracker performance compared ground-truth tests is performed and against state-of-the-art gaze estimation methods.

Finally, **Chapter 8** presents a summary of the technical achievements presented in each of the main chapters.

1.3 Relevant Publications

A. Vicente, J. Liu and G. Z. Yang, *Surface classification based on vibration on omni wheel mobile base*, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 916-921

1.4 Research Contributions

Overall, this thesis proposes a number of improvements on the mobile assistive technology. Firstly, in the context of the intended hospital and healthcare environment where the outlined technology would be deployed, the hardware innovations are few but significant. While other individual research focuses on isolated aspects of the development of assistive robotics, this thesis combines multiple fields (e.g. mechanics, electronics, mapping, navigation object-detection and gaze-tracking) synergistically, to create one unitary prototype which can then be used to conduct analyses on various aspects of software development. As such, this thesis serves to demonstrate how it is possible to integrate a multidisciplinary solution into a single robotics solution. The contributions of each of the core chapters are as follows:

- **Mobile Robot Hardware Design:** A complete working prototype for the envisioned assistive robot is comprised of a circular omni-directional base with two Lidars, depth sensors, two on-board computers, a self-contained battery with its own recharging system, and a passive suspension. The latter ensures that all four wheels maintain contact with the floor at all times – a requirement for using more than three omni-wheels effectively.
- **Floor classification:** A new method for classifying floors using vibration-only measurements generated by the rotation of the omni-wheels at the base. Previous floor-classification methods only covered classification using rubber wheels.

- **Dynamic Mapping:** An advance in dynamic mapping methodology which is achieved by combining two algorithms, a state-of-the-art SLAM techniques and machine learning-based leg detection, in order to create a novel method for accurately mapping 2D environments with people, whether there are moving or stationary.
- **Dynamic Navigation:** A six-fold increase of processing speed for real-time path planning which is based on current state-of-the-art dynamic path-planning methods, modelling a real-world situation of deployment in an actual populated environment.
- **Sensor Fusion for Long-Range Gaze Estimation:** A sensory fusion solution that increases the accuracy of long-range gaze direction estimation by combining three-dimensional sensing with hardware-based IR eye-tracking.

Chapter 2

2 Background

2.1 Introduction

The role of robotics in healthcare originated as tools for surgery as well as rehabilitation exercises and assisting with everyday activities. Today, robotics can be seen in multiple areas of medicine, ranging from surgery to therapy. Six of most influential areas of robotics in healthcare [7] could be divided into two groups, according to the nature of their Human-Robot-Interaction (HRI). The first group is comprised of intelligent robotic tools, which can either function on or in the body autonomously, in the case of intelligent prosthesis and endoscopic medical capsules, or be operated by an expert during a procedure or therapy session, as in robotic assisted surgical procedures and motor rehabilitation sessions. The second group consists of robotic platforms capable of interacting with the patient as an independent entity. They are able to perform various functions via receiving and interpreting human speech, visual cues and sometimes touch. These robots can be driven remotely, like the telepresence robots used by doctors for monitoring patients remotely, or autonomous, in the case of cognitive therapeutic robots.

Robotic Tool	Robotic Agent
<ul style="list-style-type: none">• Smart endoscopic medical capsules• Robotic assisted surgical instruments• Rehabilitation therapy• Intelligent prosthetics	<ul style="list-style-type: none">• Patient monitoring and support• Assisted mental, cognitive and social therapy

Table 2.1 – Seven of the major areas of healthcare robotics divided into two groups: Robotic Tools and Robotic Agents

Another broad research area that is closely related to healthcare robotics is the AI assisted diagnostics. This area has the potential to be integrated into autonomous robotic agents in order to detect possible abnormal conditions on the patients and alert the necessary staff. In the past, machine learning techniques, such as Bayesian networks, have been used for diagnosing diseases based on patient data

collected by the clinician [8]. Today, after the advent of deep learning methods, AI assisted diagnostics is now being used in medical imaging, for example in detecting different diseases in retinal scans [9].

The focus of the research presented in the thesis, however, is at improving the autonomous capabilities of current state-of-the-art agent-type robots in order to alleviate the workload of hospital staff. The first type of autonomous agents evaluated were social and cognitive assistive robots. This is a relatively new field in robotics where research started appearing in the literature throughout the 1990s. Whilst it continues to be developed, this field predominantly focuses on various forms of Human-Robot-Interface (HRI), which lent themselves particularly well to treating individuals who live with cognitive impairments such as autism and dementia.

A Socially Assistive Robot (SAR) tends to have an anthropomorphic or animal shape in order to facilitate interaction with the patient. One of the first SARs was dubbed *Paro*, a seal-like robot introduced in the late 1990s that can react to touch, light, sound, temperature and detect how it is being handled [10]. To this day, *Paro* continues to assist with individual and group therapies of elderly patients.

In the 2000s, a new generation of SARs with anthropomorphic features was introduced. Notable mentions include *Bandit*, an anthropomorphic robot comprised of a head, torso, and two arms on wheels that can talk back to the user and demonstrate a variety of facial expressions [11]. Other SARs were also designed to assist with communication therapies, for example *Silbot* and *Mero* [12], which were equipped with wheels, an anthropomorphic face capable of displaying various expressions, and a screen used for telepresence in remote therapy.

In the 2010s, new SARs were created to help people with dementia. These new robots included the *JustoCat*, a cat-like robot similar in size to a real cat and capable of purring, meowing and responding to stroking [13], and doll-like SARs like Nodding Kabochan, a child-like doll robot that can sing, nod and play verbal games [12]. With advancements in actuator miniaturisation, small humanoid robots were also designed to aid children with autism and other social-skills impairment, like *NAO*, a fully

articulated 58 cm humanoid robot capable of walking, talking and interacting with people autonomously.

The SARs represent an important role of robotics in healthcare as they not only help the patient to cope with his/her disease but also provide insights over long periods of time on the cognitive state of the person to the specialists. The social and cognitive assistive field requires expertise and is often a tailored solution for each patient (regarding which type of robot to use and what settings to enable).

Simultaneously, another strand of development was emerging. To facilitate patient monitoring, guiding visitors and delivering goods by means of agent-type assistive robots, prototypes were created to facilitate various forms of human interaction in health and social care environments, unlike the therapeutic environments in which SARs tend to be deployed.

In 1998, *MINERVA* was designed as a museum tour-guide assistive robot [14]. Equipped with sonars and an upright camera, it was able to navigate amongst people and localise itself in the museum regardless of crowds moving around it. In 2002, *Pearl*, another prototype, was designed as an assistive robot situated in the setting of elderly homes. It was able to guide residents and visitors around the building, whilst also delivering medicine, reminding patients of their tasks, and assisting them with safe transfer to chairs and beds [15]. Similarly, in 2011, *Care-o-Bot 3* was designed to assist elderly patients with their everyday routines. Most of its workflow processes involved delivering medicine and small objects to and from tables using a robotic arm. It could also use its tablet-tray to play games that exercise the memory and mental health of the patients. *Care-o-Bot 3* was a great improvement when compared to previous assistive robot prototypes.

The ensuing years were marked by the introduction of telemedicine robotic. For example, *InTouchHealth* manufactured two prototypes specifically designed for hospital environments in rapid succession. Firstly, the *RP-7* [6] was launched in 2012. A year later, its successor *RP-Vita* [16] resulted in a set of improvements. What these robots permitted was the ability for doctors to visually examine a given patient from a remote locale, whilst also supervising nurses and interns on duty. The remarkable success of the RT series ignited a very real interest in robotics as a means for providing telemedicine,

in which a small team of doctors can remotely monitor a portfolio of hospitals simultaneously. An honourable mention is the humanoid robot Peper, designed by SoftBank Robotics Corp. and Aldebaran Robotics SAS and launched in 2014 [17]. Pepper was originally designed to be an entertainment-type robot; therefore, it is lightweight (28Kg); it is equipped two arms for performing gestures, an anthropomorphic head, a touch screen on its ‘chest’ and a holonomic based to allow it to easily manoeuvre freely through crowd. Pepper has also been used as a tele-presence educational agent to teach new languages to children through interactive lessons remotely [18]. Figure 2.1 shows a timeline of some major mobile assistive robots from recent years.

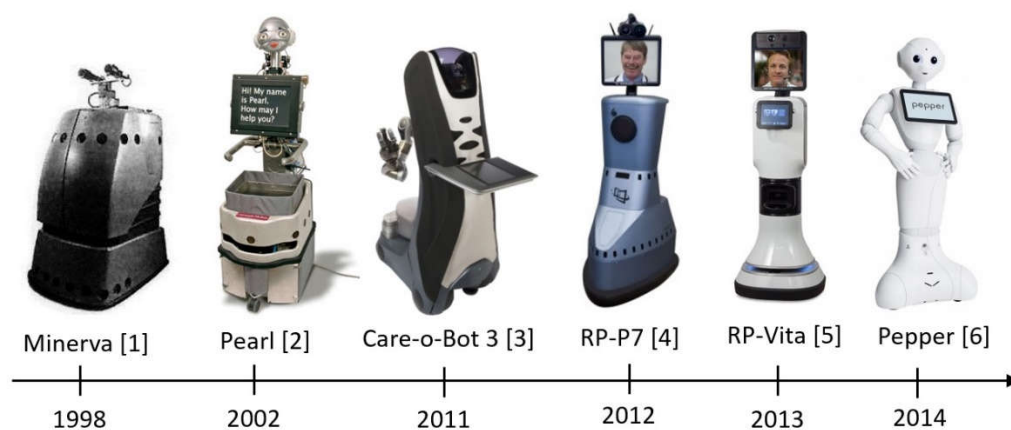


Figure 2.1 – Mobile assistive robot evolution over the years. From left to right, Minerva, Pearl, Care-o-Bot 3, RP-7, RP-Vita and Pepper.

Largely due to the tripartite influence of cheaper components, 3D printing, and the accessibility of online information, the need for a common development platform became an urgent priority in the decade leading up to the present. In response to such influences, a collaborative project between the Willow Garage and Stanford University brought to fruition a fully open-source operating system, and one which was amenable to use in most types robots. The 2007 launch of what is known as the Robot Operating System (ROS), made it much easier and accessible to build and test robots without having to ‘reinvent the wheel’ each time a new robot platform was created. As a meta-operating system

¹ Image from <https://www.cs.cmu.edu/~minerva/press/nyt/nyt.html>

² Image from <https://www.cmu.edu/cmtoday/issues/dec-2004-issue/feature-stories/human-health/index.html>

³ Image from <https://www.care-o-bot.de/en/care-o-bot-3/download/images.html>

⁴ Image from <https://www.roboticstoday.com/robots/rp-7>

⁵ Image from <https://intouchhealth.com/>

⁶ Image from <https://www.softbankrobotics.com/emea/en/press/gallery/pepper>

running on Linux, ROS provides drivers for direct hardware interaction, tools for development of new robotics platforms, and libraries that permit the sharing algorithms and solutions.

The Robot Operating System (ROS) heralded what amounted to an entirely new phase of robotics research [19]. This is because ROS permits developers, researchers and hobbyists alike to start at a level playing-field in terms of hardware support whilst building upon the collaborative work of hundreds of active contributors that continue to shape and evolve ROS. At the time of this research, ROS has proven central to the support and development of numerous prototypes and commercial products, serving as the most popular open-source platform for robotic development in both research and industry centres.

2.2 Mapping and Localisation in Mobile Robotics

One of the earliest problems encountered in mobile robotics was that of creating a map of a given environment whilst simultaneously locating itself within it. The common term used to describe this problem is known as Simultaneous Localisation and Mapping (SLAM).

There are many different types of SLAM, each designed for a particular set of conditions, but they all have the goal of locating the robot within a set of known references. As the robot moves into unknown areas, new references are added and related to the previous references, as demonstrated in [20]. In 2D mobile robot mapping, most SLAM implementation uses occupancy grids [21] to represent a finite area around the robot, whose individual cells represent the probability of being empty or not. As the robot navigates and acquires more samples from the environment, the occupancy map is updated as well as its own global position relative to where it started.

The complexity of the SLAM problem is defined by factors such as the type of sensor, the number of sensors used for fusion, and the size and/or resolution of the environment to be mapped. SLAM-based methods usually rely on the Kalman filter [22, 23] and/or a particle filter [24-26] to keep track of the robot's whereabouts by combining multiple acquired samples into a representation of the environment. The relative location of the robot is calculated by the degree to which the samples acquired by the sensors match the previously-seen samples of the environment.

However, purely Kalman-based SLAM methods often run into performance bottlenecks as the array of samples increases. This impediment is an inherent characteristic of Kalman-based methods in that every newly-acquired sample must be checked with previous samples in order to estimate a Bayesian posterior probability of the match. In turn, particle-based methods have the limitation of requiring hundreds of particles to simulate sampling of the map in memory. Although more capable of localising the robot when initialised at a random location, this method requires considerably more processing power. This is because it is necessary to update all particles at each sensor update, which can require the detection of thousands of data-sets in the case of complex environments.

Sonar-based SLAM was one of the earliest forms of SLAM, where arrays of sonars were mounted around the robot's chassis and fired in sequence in order to measure the approximate shape of the environment [27, 28]. Due to their inherent limited resolution and multipath echo characteristics, the use of sonars in mapping could not produce accurate maps. That said, they served to aid the development of the inverse sensor model [29], which allowed a more accurate approximation and faster computation of maps. As such, today, sonars are no longer used for mapping; instead, they are deployed, in mobile robots, to complement the use of cameras and light-based distance-sensors when measuring obstacles in order to detect glass walls.

With the introduction of Lidar sensors, modulated IR laser distance finders, SLAM method became more accurate due the new sensor's increased accuracy and low latency while sampling. Notable Lidar manufacturers, Hokuyo and SICK, are currently dominant in the market. Lidar-based 2D SLAM techniques, such as GridSLAM, FastSLAM and Gmapping [24-26], rely on a large number of samples per scan and on the robot's odometry, estimated by the encoders, to constantly update the map and the robot's position. Recent SLAM techniques such as the ICP-SLAM [30] are able to keep a constant estimation of the robot's odometry by matching previous and current scans as the robot moves around. Unlike particle methods used in previous SLAM implementations, HectorSLAM [31] uses an iterative method to align the new scan with the current map. These SLAM methods so far were designed based on the assumption that the environment is static or sparsely populated, which simplifies the problem of

mapping. When considering moving obstacles, tracking must be taking into account, like in SLAMMOT [32], a method developed to undertake dynamic environments as it adds tracking to detection of reference points and is able to differentiate moving objects from walls and static objects.

The challenges of mapping environments in three dimensions were both computationally and resource-wise unviable for many researchers until approximately 10 years ago, when 3D depth sensors became cheaper and mobile processors started to process stereo-vision in real time. Some of the first depth cameras used Time-of-Flight technology, like the MESA SR3000, to create a 3D estimation of the scene in front of them. They were as expensive as high-end Lidars today and had very low resolutions.

In the 2010s, sensors like the Microsoft Kinect [33] and the Asus Xtion Plus were introduced in the consumer market; signalling the beginning a new generation of low cost depth sensors. These sensors use structured lighting to estimate the depth of a scene by projecting IR dot patterns onto the environment which are then detected by an IR camera; their distortion is correlated to the distance. These depth sensors became very popular among hobbyists and robotics researchers alike, including being present in final products such as the tele-presence robot *RP-Vita* [16]. Structured-lighting sensors, however, tend to perform poorly in outdoor environments, due to the extra IR light from the sun, and have a limited range, between 3 to 5 meters. Autonomous cars and other types of large mobile robots have more reliable 3D depth sensing that can more than 100 meters ahead in order to give the system enough time to detect an emergency situation and react in time. One solution is using multi-channel Lidars, such as the ones produced by Velodyne [34]. These Lidars cost four to five times as much as the single channel 2D Lidars but possess as many as 64 independent lasers, in the case of the HDL-64E, which is capable of generating a 360-degree point clouds of the environment with up to 120m range.

Vision-Based SLAM is a more complex problem, as it deals with estimating 3D position and orientation from a 2D image of the world. Many different solutions for this mapping problem have been proposed for both indoor [35-38] and outdoor environments [39] with variables on both mono and stereo implementations. More recently the introduction of omni-directional camera lenses allowed for a 360

degree visualisation and mapping of the environment with relative reduced hardware specifications [40]. As the main method for mapping and localisation employed in this research is laser based, the exploration of more topics in vision-based SLAM is outside the scope of this thesis.

Method	Advantages	Disadvantages
GridSLAM	Pioneer method using occupancy grid for localisation.	Slow and complexity for feature matching grows almost exponentially
FastSLAM	Improvement over the GridSLAM with efficient feature association	Still requires large memory footprint for storing all features
GMapping	Improvement in pose estimation by combining the robot's movement and particle observation	Not accurate for large scale mapping
ICP-SLAM	Produces odometry by matching two scans	Can produce larger drift over time
HectorSLAM	Mapping and localisation achieved by aligning new scan with current occupancy map	Cannot accommodate dynamic objects detection
SLAMMOT	Can track moving objects by introducing uncertainty states for elements that may be dynamic on the map	Cannot differentiate dynamic objects if they stand still

Table 2.2 – Summary of relevant SLAM techniques

2.3 People detection

The mobile assistive robot envisioned in this research must be able to locate people in a crowded environment in order to avoid them as it moves towards its goal. The method for detecting people directly relies on what sensors are available on the robot platform. Cameras are a very popular choice of sensors because of their low price and versatility in terms of shape detection. Haar features [41] and Histogram of Gradients (HOG) [42] are two of the most common set of image features used for classifying faces and bodies in images, respectively.

Haar features are used with a cascade classifier that is trained with many positive and negative samples. These samples include edge, line and corner features that compose the object of interest in the image, and in this case, a person's face. Each feature within a sample window is the result of the difference between the sums of various regions of the sample window.

The HOG detector requires far more operations to be calculated compared to the Haar features but is able to provide values that more accurately describe the direction of the edges of a sample. For a

sample patch, usually an 8 by 8 patch, a histogram of edge direction between 0 and 180 degrees is created and later on normalised. These features, like the Haar classifier, are used to train a binary classifier to detect the objects of interest. The main disadvantage of image processing methods in general, however, lies on the relatively high requirement for processing power in order to run the detector in real time.

Unlike image-based methods, Lidar-based people detection can more accurately estimate someone's position relative to the robot, but only have a fraction of the amount of data image-based methods to reach the same results. Some of the earliest popular methods for detecting legs in a Lidar scan were based on the geometric arrangement of the points. The work published by Xavier et al. [43] demonstrated how legs could be extracted by identifying arcs with a radius between 0.1 and 0.25 metres.

A more robust approach for leg classification was later proposed by Arras et al. [44] using boosted features for classifying a wider range of leg segments. The 2D scan is split into segments according to the distance between their consecutive points. If two points are further away than 10 cm, a new segment is created. Next, geometric and statistical features from each of the segments, such as linearity, circularity and mean distance between points, are used by an Adaboost classifier.

The use of multiple layers of Lidars has been proposed as an attempt to increase the people-detection accuracy by providing a more three-dimensional data format. A method using two pairs of Lidars positioned at different heights on the robot (knee and torso heights of an average adult) was proposed by Carballo et al. [45]. Two overlapping Lidars were set to guarantee 360 degrees of coverage, and both torso and knees are associated to classify people standing and walking in the environment.

More recent methods for people detection are based on long distances using multi-layer Lidars, like the Velodyne 3D Lidars. These Lidars, originally designed for autonomous vehicles, can contain 16, 32 or 64 lasers that can reach up to 60m in all directions. In the work presented by Shackleton et al. [46], people are detected by removing the ground plane and segmenting the remaining point cloud into clusters by proximity. SPIN images are then created for each sub-cluster and used to train the classifier. A more accurate classification method was presented by Spinello et al. [47], where instead of extracting

the SPIN images from the remaining sub-clusters, each sub-cluster was sliced into 2D clusters at specific heights. 2D features (e.g. boundary length, width, circularity) are extracted from each layer and concatenated into one single feature vector for that cluster.

Hybrid methods that combine Lidar and cameras were also explored, for example, in an image and shape pattern classification of legs in scans which was presented by Bellotto et al. [48]. The detector looks for three distinct shape signatures representing both legs visible, one leg visible, or one leg visible with the other partially visible. The system also uses a camera facing the direction of the Lidar sensor and associates any faces detected by the Haar filter with the leg candidates found by the Lidar algorithm.

One of the most recent and popular methods combining both RGB and depth (RGBD) information for detecting people was presented by Munaro et al. [49] and further developed in [50]. Like the 3D Lidar methods, the detection also removes the ground plane and separates the remaining clusters in the point cloud by distance. The HOG for each cluster is calculated and used to detect people. The tracking, also present in the detector, uses the histogram of colour information of each of the clusters to track their movement in the scene. This detector is readily available in ROS and is widely used in the robotics community. It does, however, require a powerful processor (i5, i7 or Xeon) to run it in real time and has a range limitation of around three metres for accurate detections.

Method	Sensor type	Summary
Haar	Camera	Binary cascade classifier trained on positive and negative image samples like lines and textures.
HOG	Colour Camera	Similar to Haar, but uses 8 x 8 patches to calculate mean gradient orientation
Xavier et al. [43]	Single 2D Lidar	Legs are detected by matching arcs to scan segments
Arras et al. [44]	Single 2D Lidar	Adaboost classifier trained on geometric descriptors of consecutive scan segments
Carballo et al. [45]	Multi 2D Lidar	People are detected by matching torso and leg scan segments
Shackleton et al. [46]	3D Lidar	Floor is removed from 3D point cloud leaving clusters representing people and other objects which are converted to SPIN images for classification
Spinello et al. [47]	3D Lidar	Clusters of points left after the floor has been removed are sliced into 2D segments and classified as a group to detect the person
Bellotto et al. [48]	Colour Camera and 2D Lidar	Leg detection and face recognition are used to confirm when a person is detected
Munaro et al. [49]	Depth Camera and Colour Camera	Point cloud processing is combined with the colour histogram of clusters in order to more accurately identify individuals based on their shape and appearance

Figure 2.2 – Summary of relevant people-detection techniques

2.4 Navigation

The new assistive robot is expected to find its way to the desired location in any known map and be able to manoeuvre through any moving obstacles in it. As the robot itself can only manoeuvre on a plane, only 2D navigation is considered in this thesis. Navigation can be divided into global and local planning, which are both illustrated in Figure 2.3.

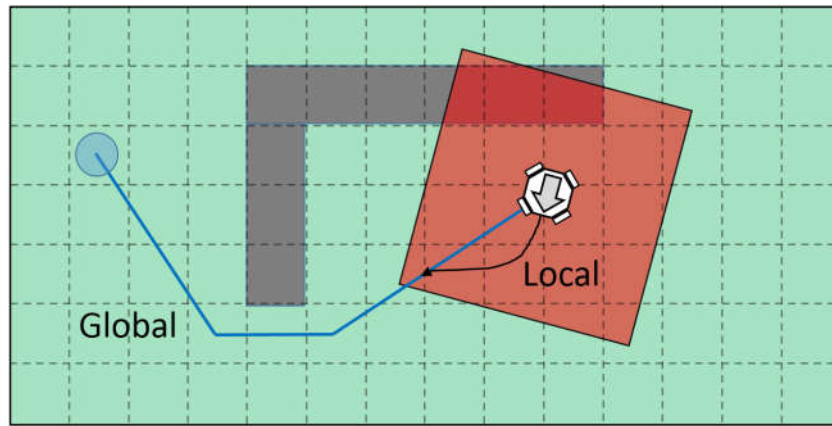


Figure 2.3 – Representation of global and local planning on a map scale. The green squares represent all available free space where the robot can move. The grey squares represent obstructed space, such as a wall. The red area around the robot represents its local planning space.

2.4.1 Global Planning

Global planning describes the process of calculating the path, or paths, that connect the robot's current location to its target location. In path planning, the path between the robot and its target location is represented by nodes on a graph, where the graph represents all known locations on the map. The edges on the graph represent the possible transition to and from different locations. In the case of occupancy grids, nodes are the cells on the grid, where each cell is connected to its immediate neighbours, and the graph is the grid itself.

The Dijkstra algorithm is one of the simplest graph transversal techniques. It employs a depth-first search strategy, which is better than brute force, and is certain to attain the shortest path; however, it is still very inefficient for 2D occupancy grids.

The A* algorithm is a well-known technique for finding the shortest path in a graph as it outperforms Dijkstra by using a heuristic search that prunes the search area as it moves through the graph. The heuristic employed in an occupancy grid minimises the current distance between the robot and the target. A* is also guaranteed to find the shortest path, provided that there is at least one path connecting the robot to its target.

The Theta* [51], also known as “any-angle path planning” is yet another variation of the A* algorithm which reduces the minimum path by connection visible nodes rather than obeying the normal 8-way route on a grid. It represents an improvement of the Field D* [52] method, as it has similar complexity to the original A* method, but with an overall shorter path.

The previous three methods mentioned were designed for static environments, but in order to account for constant modification of the routes, new conditions had to be introduced during the planning phase. An example is the Safe Interval Path Planning (SIPP) [53], which introduces time intervals when paths are blocked in order mitigate the constant need for route re-planning. The algorithm uses A* for the global planning and an estimated trajectory of the person is calculated to detect possible collisions. If a collision is detected, the robot uses a time interval to wait for the person to move and re-evaluates the viability of the original planned path. An extension of this method that allows “any time” action, rather than only during intervals, was presented by [54].

A graph-based method for re-planning the global paths was introduced in [55]. It uses people’s common routes in an environment to create nodes on a graph representing the most likely places to avoid and follow. The robot initially follows the human instructor around the environment observing what routes they take and assigning nodes to all locations visited by the instructor. Once the initial tour is complete, the graph is optimised and the robot is ready to navigate. During normal operation, A* is used to find the shortest path most likely used by people using the graph. The local placement of the nodes visited by the instructor also ensures that the robot does not bump into obstacles not previously detected by its sensors.

Geometric approaches for global path planning were also explored, for example, with the chance-constrained optimisation introduced in [56]. This method uses a series of convex obstacles rather than an occupancy grid. Concave obstacles are represented by multiple convex shapes arranged together. The path planning is done using a third-party software with a proprietary algorithm. Other noteworthy methods include the use of Gaussian Processes (GP), proposed in [57], to model people's trajectories while moving in a dynamic environment, and the use of Rapid exploring Random Tree (RRT), for concatenating all possible paths. A variation of the same method was presented in [58] using a Chance Constrained RRT (CC-RRT) for the planning phase.

Variations of classical global planning methods also include the introduction of Machine Learning, for example, by [59] which uses Inverse Reinforcement Learning (IRL) to learn how to move through the environment from common paths taken by humans. As IRL requires all variables of all samples to be known during the training phase, Gaussian Mixture regression models [60] were used to estimate unknown or uncertain values, such as the number of people in the crowd and each person's velocity, that could not be seen at all times inside the sensor's field-of-view. A summary of all presented methods can be found in Table 2.3 below.

Method	Environment	Summary
Dijkstra	Static	Extensive search with complexity $O(n^2)$ on 2D occupancy grids
A*	Static	Guaranteed to find shortest distance within node constraints
Theta*	Static	An "any-degree" approach to the A* method which yields shorter paths and doesn't follow node constraints
SIPP	Dynamic	Introduces safe time intervals to wait for people to move around before resuming navigation
CC-RRT	Dynamic	Enables more accurate state prediction of moving obstacles by simulating them prior to executing subsequent manoeuvres
GP based	Dynamic	Describes a more fluid path prediction for people by using Gaussian Processes
RL based	Dynamic	Learns how to better navigate based on trial and error

Table 2.3 – Summary of relevant global planning methods

2.4.2 Local Planning

Unlike global planning, local planning focuses on how to avoid and manoeuvre around immediate obstacles surrounding the robot. Local planning is necessary to ensure that the robot can adapt to small environments changes, such as a moved object on the floor or even people walking around.

Potential field-based methods, Khatib et al. [61], assign a repulse vector field to obstacles and an attractor point to the robot's goal. The robot itself behaves a particle on the field and its target velocity is dictated by the vector sum of all nearby field gradients. This method, however does not take into consideration the robot's own dynamic constraints and may lead to the robot being stuck on a local minima in the potential field.

The Vector Field Histogram (VFH) method) by Borenstein et al. [62], presents an alternative to obstacle avoidance by creating a polar histogram of sampled distances measured by the laser range finder. In this histogram, empty spaces are represented by high peaks and obstacles by valleys. The algorithm calculates the robot's heading and speed by detecting the position and size of the optimal peak on the histogram. This method also suffers from local minima, but variations of the VFH which try to remedy its short planning phase by adding a further exploration step, Ulrich et al. [63].

Compared to the previous methods, the Dynamic Window Approach (DWA) by Fox et al. [64], demonstrated higher stability by introducing the robot's locomotion constraints while calculating its velocity. The DWA effectively creates a velocity control space where only admissible velocities, I.E. velocities in which the robot is certain to reach its destination within the time window, are considered. Variations of the DWA also incorporated navigation functions in order to predict the optimum path on a larger scale than local planning [65-68]. The DWA was not originally designed to handle dynamic objects moving around the robot, therefore several techniques have been developed [69, 70] to overcome this limitation. One drawback of the DWA is its overshooting characteristic when the robot increases speed and must make a sharp turn. Since the resulting speed of the robot is calculated using vectors, one vector can bias the robot's final direction. The Nearest Diagram (ND) by Javier et al. [71],

was a new geometric algorithm introduced in that used reactive and physical-based rules to avoid obstacles around the robot.

The previously mentioned methods work well on guiding the robot around unexpected static obstacles, however, when dynamic obstacles are introduced, such as walking people, a collision can occur; hence, a different set of rules must be used to take the obstacle's velocity into account. The Velocity Obstacles (VOs) by Fiorini et al. [72], were introduced as areas inside the robot's configuration space that should be avoided in order to prevent future collisions. The VOs are calculated using the position and velocity of both the robot and obstacles in the immediate surroundings. The algorithm described in [72] selects an action that will move the robot closer to its final goal while maintaining a safe distance from the VOs. An improved version of the VO algorithm was later presented by Gal et al. [73], where a time limit is used to ensure that the robot stops before any collisions happen.

Another type of local navigation strategy called the Elastic Band method was introduced by [74]; it used a finite number of circular areas within the robot's configuration space to quickly change its trajectory when unexpected obstacles are encountered. The trajectory can be re-traced to any other trajectory that is contained in the pre-determined circles in the configuration space.

The Inevitable Collision States (ICSs) method, introduced by Fraichard et al. [75], tackled the collision problem within the robot's configuration space by delimiting and avoiding areas that denote immediate collision, similar to the VO method. If the position and velocity of all moving elements within the scene are known, the robot will not crash into anything as long as its trajectory does not cross an ICS. In practice, due to the limitations of the sensors, not all obstacles can be detected at all times thus some collisions can still happen. The Breaking ICSs was introduced by Bouraine et al. [76] as a way of mitigating collisions where both the robot and the person are moving was proposed. This method takes into consideration the limitations of the sensors' field-of-view when predicting future collisions and ensures that the robot completely stops prior to any collision with a moving obstacle.

Probability collision states (PCSs) by Althoff et al. [77, 78], were introduced as an alternative method for including obstacle uncertainty into the trajectory planning method of ICS. During the

planning phase, the probability of collision is used as a cost function. The robot's next movement is selected by sampling the cost for different possible movements for every situation and selecting the lowest cost. A simulated cost for all moving obstacles' movements is also added into the process to maximise the chances of finding an optimal solution. Another benefit of adding simulated object movement is to avoid conditions where the robot's own movement may force a person into a collision.

Method	Advantages	Disadvantages
Potential Fields	Uses Maxell's potential field equations to create a reactive obstacle avoidance based on the repulsive forces from the obstacles and attractive force of the goal	The robot can get stuck in a local minimum created by a concave arrangements of obstacles.
VFH	simple to implement as it creates a histogram of scan measurements representing empty spaces where the robot can go	This method also suffers from local minima, allowing the robot to get stuck in between obstacles
DWA	Uses the robot's kinematics constraints to calculate possible velocities for the base within a sampled window from the environment	Tends to overshoot when the path takes a sharp turn, like rolling into a room from a perpendicular hallway
Nearest Diagram	Employs reactive and physical-based rules to avoid obstacles around the robot in order to avoid kinematic "traps"	Not straight-forward to port the control from robot to robot, as it must be tuned and configure to each robot's kinematic model
Velocity Object	Creates avoidance areas in the robot's planning stage that represent immediate collisions with incoming moving objects	Requires more processing power for the cost function to run for each of the detected obstacles
Elastic band	Using circular areas along the global path, the robot re-calculates its local trajectory within these safe areas to get to its destination	Not always guaranteed to reach optimal path, and usually used in conjunction with other path planning methods
ICSs	Similar to VOs, ICSs define areas that must be avoided to avoid immediate collision	Requires accurate detection of all elements within proximity, including those out of sight
Horizon Control	Introduces a short simulation time before making decisions that reduces the possibility of collisions in the future	Higher processing power requirements

Table 2.4 – Summary of local planning methods

The methods described by Toit et al. [79, 80] used a closed-form approximation of the probability distribution of the moving obstacle's position and velocity while evaluating its trajectory. This closed-form term allowed a fast calculation of the probability of collision given a specific trajectory. A receding

Horizon Control scheme along with a chance constrain optimisation were used to find the optimal trajectory. A probability of collision threshold is used to determine whether a trajectory is safe or not.

Another solution for the obstacle avoidance problem was proposed by Tychonievich et al. [81], in which detecting collisions is achieved by finding the roots of a polynomial function. The robot's physical constraints and movements are approximated into Chebyshev polynomial functions and RRT is used to find the manoeuvres which will lead to a collision-free trajectory.

2.5 Environment Perception

In addition to being able to map the environment in 2D and 3D, mobile robots can also be used to detect attributes of the environment that may not be identifiable by the naked eye, for example, the presence of high levels of carbon dioxide in the air, abnormal room temperature or even radiation leakage. Periodic inspections by these autonomous robots would help identify possible health hazards that otherwise would be left unchecked. One of the focus areas of this research is in identifying floor types where the assistive robot will operate in order to better adjust speed for noise levels and detect possibly damaged floor patches.

The introduction of the Pioneer 3-AT [82] off-road robot base, equipped with four rubber wheels and differential drive configuration, allowed many research groups to explore ground classification in outdoor environments.



[1]

Figure 2.4 – The Pioneer 3-AT robot platform; used in different research on outdoor surface classification

Previous floor-classification methods on mobile robots mainly relied on robots equipped with rubber tires to collect the data, which dampens the vibration due to soft contact of the wheels [83-85].

¹ Image from <https://www.generationrobots.com/en/402397-robot-mobile-pioneer-3-at.html>

An omni-directional robot equipped with omni or Mecanum wheels would introduce extra noise due to the contact points of the rollers while the robot is moving. This would, in theory, significantly reduce the signal-to-noise ratio of the vibration signal from the floor, thus increasing the difficulty of this task.

Over the years, different methods for classifying surfaces were developed based on different sensing technologies. One popular method is to use accelerometers. The vibration created during the movement of the robot over a surface can be measured as a one-dimensional acceleration signal perpendicular to the surface over time. Such signals can be further processed for feature extraction and surface identification. An accelerometer attached to a “tail probe” touching the ground can provide clear enough features from vibration to efficiently classify multiple surfaces [86]. One advantage of this method is that the signal measured relies on the direct contact to the ground texture. However, it also hinders the robot’s range of motion and creates undesired noise from the probe dragging on the floor.

A different approach requires the vibration sensor to be firmly attached directly to the chassis of the robot in order to measure the vibration on the robot’s body. Most works using this method focused on off-road robots [84, 87] equipped with rubber wheels and moving on surfaces such as grass, gravel or asphalt.

A variant of this method introduced a speed invariant classification [88] but with poor accuracy. The chassis-mounted sensor method is ideal for most scenarios as it does not interfere with the robot’s design, nor introduce extra disturbances to the environment; however, the vibration measured by the sensors is greatly dampened by the mass of the robot’s body itself, effectively reducing the signal-to-noise ratio of the texture vibration.

Microphones have also been used to sense the sound generated by the wheels while the robot is moving to classify the type of surface. In the work presented by Roy et al. [89], a boom microphone was mounted near the wheels and pointing downward. Despite its high classification rate on three different floors, this method is sensitive to external sound sources, such as ambient noise, which could bias the final classification results.

One of the early steps in the development of tactile sensors for texture recognition was inspired by the human finger. Embedded strain gauges were used to measure the mechanical stress of the sensor generated by friction when moved over a surface [90, 91]. Further development led to the use of sliding movements as a way to characterise surfaces with more precision [92]. One of the latest methods for texture-classification employs a multi-sensor-equipped artificial finger that is capable of detecting different surfaces via vibration and elasticity of the fingertips [93]. These methods were aimed at small controlled environments, where the sensor was always placed at an ideal position and with ideal pressure over the texture to be classified. These conditions are rare to non-existent in a normal, real-life scenario with a mobile robot.

2.6 Human-Robot-Interaction (HRI)

Human Robot Interaction (HRI) is the area of cognitive robotics that studies the interactions between humans and robots in different roles and tasks. HRI is a relatively new field in robotics research that only began real development in early 2000s [94] and is a successor to Human Computer Interaction (HCI), which started more than 40 years ago [95].

There are, however, key differences between HCI and HRI. Unlike computer platforms, assistive robots require a physical understanding of the world in order to interact with it and the user. Robots also present a certain degree of uncertainty, unlike simulations, that prohibit two identical robots to perform the same tasks identically. This includes nonlinearities of sensors and mechanical failure/wearing of actuators [96]. Robots also can perform different roles when interacting with humans and other robots, which is restricted from human-computer interaction [94].

2.6.1 Interaction roles

It was proposed by [96] that humans and robots interact in five different types of roles, where a human may carry more than one role simultaneously and multiple humans may interact with the same robot. However, for most applications of assistive robotics, only one user interacts with one robot at time, therefore this is the scenario assumed for all interactions. The roles proposed by [96] are:

- **Supervised:** The human oversees the robot teammates and requires an overall model of the current state of the task.
- **Operated:** The human directly operates all robot's functions (often referred as "telepresence") or issues goals depending on the level of autonomy of the robot.
- **Teammate:** The robot is treated by the human as part of the team and goals and responsibilities are shared.
- **Mechanic:** The human has to diagnose and fix, if possible, any problems that occur with the robot in the given task
- **Bystander:** No direct relationship between the human and the robot. Both co-exist within the same environment carrying out their own tasks.

2.6.2 Robot safety

Safety is always a concern in Human-Robot interaction as in any other environment with humans and machines. The most common, and perhaps most reliable, way of preventing human injury when interfacing with a robot is by implementing workspace segregation [97, 98].

Safety Strategy	Advantages	Disadvantages
Human pain threshold limitation	<ul style="list-style-type: none"> • Allows continuous interaction between human and assistive robots even under pressure from the human 	<ul style="list-style-type: none"> • Threshold has to be reset for the elderly and individuals with lower pain threshold • In case of software failure, joint becomes non-compliant
Force detectable artificial skin	<ul style="list-style-type: none"> • Allows precise pinpoint of the contact area along with amount of pressure • Preferable way for jointing both haptic signal perception and safety detection using one type of sensor 	<ul style="list-style-type: none"> • Skin must cover all movable parts of the robot, which can require careful coating and large areas of artificial skin may be required • Safety measure is only triggered after contact. Delay in reaction from the robot may cause human injury
Controllable compliant joints	<ul style="list-style-type: none"> • Compliance allows for a safe environment when in physical contact with humans • Different levels of compliance can be set for different performances and level can be changed on the fly 	<ul style="list-style-type: none"> • In case of software failure, joint becomes non-compliant
Mechanical compliant joints	<ul style="list-style-type: none"> • Compliance allows for a safe environment when in physical contact with humans • Compliance is independent from software, hence no coding error can jeopardise the safe operation of the joint 	<ul style="list-style-type: none"> • Compliance cannot be adjusted by software; hence, further control strategies are required to accurately model the compliance of the actual joint.
Back-drivable joints	<ul style="list-style-type: none"> • Allow torque consistency with the advantage of mechanical compliance when torque exceeds determined threshold with no damage to the mechanism • Compliance is completely software independent, hence 	<ul style="list-style-type: none"> • Requires extra sensors for position feedback when the safety mechanism is under stress

Table 2.5 – Safety strategy comparison table

Industrial robots are not allowed to operate when humans are within its workspace, and safety measures such as walls, fences and emergency switches are implemented to stop the robot in case a human opens the door/hatch to the robot's workspace.

In mobile assistive robotics, however, close proximity between the human and the assistive robot when the robot is operational is an unavoidable condition that must be dealt with using new safety measures. If robots are strong enough to assist limb movements and move people from one place to another, they are also strong enough to cause an injury in the case of accidental failure.

Several strategies have been developed over the years for preventing human injury while interacting with assistive robots. These strategies include pain tolerance limit [99], which allows the robot to move as long as the detected pressure does not exceed the pain threshold set in the controller, and force detectable artificial skin, where deformation on the artificial skin surface indicates points of contact with pressure information to the controller.

Compliant joints can simulated actively [100, 101], by sensing the torque applied on the arm using a strain gauge and the deflection of the position of the joint using a position encoder, thus enabling the controller to vary the torque applied by the servo on the joint accordingly, and passively, via the use of springs used between the servo axis and the load and back-drivable joints [102].

2.6.3 Gaze Detection

People often use gaze as a means to convey focus and intention to other parties during a conversation, either by looking at the person for whom the speech is meant or looking at an area/object of interest. The other parties in the conversation can then estimate the speaker's intention by where or at whom the speaker is looking. In HRI, gaze estimation can provide the robot more information about where the human participant's attention is directed.

Gaze estimation techniques involves two main processes: eye-detection, via image processing, and Infrared (IR) pupil detection, which requires extra hardware for detecting the pupils. The former only

works well on a very short range whereas the latter requires much more processing power to perform head and eye detection in real time.

The use of head orientation to increase the accuracy of gaze estimation was initially demonstrated by Kaminaga et al. [103]. Using two cameras in stereo-system configuration, they extracted the intensity and disparity of the regions corresponding to the head and eyes to make the estimation. A single camera version of the same method was later proposed by Nickel et al. [104]. Both implementations presented by Kaminaga et al. and Nickel et al. used the Viola-Jones Face Detection algorithm [105] on the intensity image from the cameras, using a fixed-size bounding box to track the person's head. More recent developments also include a more accurate estimation of gaze direction based on low resolution images by using intensity interpolation and shape matching [106].

Gaze detection was used in surveillance systems [107], where the head of the individuals was detected using HOG [42] descriptor, and tracking of each of the heads was achieved via Kalman Filtering. Gaze orientation angles are one of the equally divided forty-five-degree regions, where the region selection was done via a Randomised Ferns classifier. A mixture of HOG and average colour of the area of interest demonstrated the most accurate results. The presented work of Benfold et al. [108] tackled the same problem but on low-resolution videos. Hidden Markov Models (HMM) were used to ensure temporal consistency during the tracking due to quantisation noise.

In HRI applications where there is more than one human participant, it is often desirable to detect people's gaze direction in order to know if they are talking between themselves or with the robot. In the work by Benfold et al. [109], gaze estimation was used on a humanoid NAO robot to allow it to ignore other conversations and verify if participants were talking to it. Using the estimated head orientation, the proposed "visual focus of attention" was constantly updated using an HMM per individual, with the Gaussian distribution's mean set at the target head orientation. Results of this work demonstrated a more consistent angle estimation compared to other techniques that employed a Gaussian Mixture Model (GMM) instead [110].

Gaze estimation techniques that rely on pupil detection use the light reflected by the ocular fundus, also known as the red-eye effect (Stiefelhagen et al. [111]) to indicate the exact position of the pupils relative to the camera. A common implementation of this method requires two light sources: on-axis, where light comes from the camera's location and pupils reflect it; and off-axis, where light comes from a source away from the camera. In an ideal scenario where the intensity of the two light sources match, subtracting the images captured from the different lighting conditions results in only the pupils being visible on the final image.

The work presented by Hodgkinson et al. [112] utilised the red-eye effect on the visible light spectrum to track the person's pupils using a single camera. As the visible light was not filtered by the camera, an adaptive threshold model was used to compensate for the noise left in the subtracted image. For close-range gaze detection, around 50 cm or less, IR illumination can be used to detect reflections from both the lenses and cornea [113] if the image of the eye has enough resolution. These reflections are known as Purkinje images and can be used to provide additional reference points to improve the accuracy of the direction estimation.

The use of the first Purkinje image, commonly known as the "glint", along with the pupil reflections is one of the most common methods for short-range gaze detection applications [114-116], like tracking a person's gaze when looking at a screen. The method proposed by Zhu et al. [116] used two off-axis vertical LED arrays on both sides of the camera and required the head to be static due to its inability to be calibrated for each individual user. In order to compensate for these drawbacks, neural networks were used to create a regression model for the glint position so that gaze could be estimated for different users and even small head movements [115].

The use of two reflections for gaze detection was demonstrated by Czy et al. [117] using a head-mounted camera and IR light source. The IR light source produced the first and fourth Purkinje images, and the pupil centre was estimated using edge detection on a threshold-passed image. The depth of the eye relative to the camera was estimated by using a neural network on a few training images per user.

The final geometric position and orientation of the eye relative to the screen coordinates was then estimated using a geometric transformation.

The method presented by Lee et al. [118] utilised four Purkinje images to detect gaze estimation on a screen. An LED light source was attached to each of the four corners of the screen. Two cameras were used, one on each side of the screen, to compensate for head movement. The first camera detected the person's face using a face-detection algorithm while the second camera was set with a much narrower field of view to be able to focus on the eyes; hence, it was mounted on a pan-and-tilt base. It also had an on-axis light source to detect the pupil reflection. The four reflections, detected by the second camera, were mapped to the co-planar light sources on the screen through a calibration procedure to allow the gaze position and direction to be calculated relative to the screen.

2.6.4 Intention recognition

Intention is largely accepted as “*a plan of action the organism chooses and commits itself to the pursuit of a goal—an intention thus includes both a means (action plan) as well as a goal*”[119]. There are various approaches and methods for understanding and demonstrating intention for the assistive robot. In this topic, brief descriptions of some of the commonly used concepts in intention recognition are explained.

As proposed by Tomasello et al. [120], there are effectively three types of recognition: keyhole recognition, intended recognition and obstructed recognition. In the first type, the observed agent is unaware of the observer, and proceeds executing the plan without any special consideration for the observer. In the second type, the observed agent is aware of the observer and actively cooperates in the recognition, for example, by ensuring that crucial parts of the demonstration are not obstructed. In the third type, the observed agent is again aware of the observer but is actively trying to disrupt the recognition process and hide its intentions [120]. Humans can perform well in recognising intention in all three cases, but in assistive robotics, keyhole recognition is a more preferable scenario. Ideally, humans should be able to act naturally without worrying about explicitly demonstrating intentions to

the assistive robot when performing their actions. It is the robot's task to identify the user's intent from its fluent actions.

Recognising intention and goals from the actions performed by the human agent is a classic model-matching problem. The robot agent, via the use of sensors, measures the human agent's states at all times at a fixed sampling rate. The data collected can then be analysed using two different approaches: descriptive and generative [120].

The descriptive approach bases its assumptions on explicit information. It characterises patterns through selecting a number of low-level features and setting restrictions at the feature level, for example through Markov Random Fields [121] or deformable Models [122]. The robot agent matches the observed data against representations in its database, and according to the nature of the current task, it generates the actions corresponding to those representations. The representations in the database can be labelled as goals, beliefs and intentions to characterise their execution. This approach corresponds to the "action-effects association" method for intention interpretation [120].

The generative approach bases its assumptions on implicit information. It uses a set of hidden variables that are responsible for producing the observed data. The hidden variables represent the many degrees of freedom underlying the observed subject (human agent) and usually employ probabilistic density distributions to infer beliefs upon the observed model. In order to use these variables for a recognition process, the parameters for the probability distributions for each variable must be changed until the generated data can be easily compared against the observed data [120]. These generative models are used broadly in the machine learning field, with many variations [123, 124].

Generative recognition has been largely used in the robotics community and variations with motor constraints have also been developed [120]. The internal models created by employing the hidden variables assume the form of forward and inverse models as well as behaviours [125] and schemas [126, 127]. These models can be used as an internal simulation of perception [125], just like in the forward model representation in the Theory of Mind, which allows one to rehearse the action mentally and imply intentions on other observed actions.

2.7 Conclusions

This chapter presents an overview of related methods to the development of the next generation mobile assistive robot prototype in hospital environments. First, an overview of the major methods for mapping environments was presented, in the context of which type of sensor was used. Real-time mapping and localisation was investigated, presenting the current trends in SLAM and the advantages and disadvantages of each method for both static and dynamic environments. The most relevant SLAM methods rely on 2D Lidars and are designed for static environments. With regards to sensors, depth sensors are now much cheaper and able to provide more information about the environment but suffer from limited range and accuracy. 3D Lidars have also been recently introduced but are prohibitively expensive. One SLAM solution was found that is capable of taking into account moving obstacles, but it is unable to detect people when they are stationary. The problem of mapping an environment whilst people are in it is still an open problem.

Second, different methods for detecting people in the environment were investigated. The ability to detect people is essential for an assistive robot to plan its course without causing major disruption and recognising people as distinct from the background whilst mapping the environment. Camera-based methods demonstrated good accuracy but require significantly more processing power to run in real time compared to other methods and have a narrow field-of-view. Colour and depth cameras combined (RGBD) provide an even higher accuracy but at the cost of even higher processing power.

On the other hand, Lidar-based people detection methods require a fraction of the processing power to run, as they provide a one-dimensional array of distance values and cover almost 360 degrees of view. People detection using a single Lidar focuses on detecting legs, and some methods use Lidars at different heights to associate the torso and legs at the cost of a more expensive system. Recently introduced 3D Lidars are able to generate a 3D point cloud representation of the environment, which in turn allows for a more accurate detection, but are far more expensive than 2D Lidars and rarely used indoors. Given the detection accuracy of the methods overviewed in the literature, a 2D Lidar-based

method would be most suitable for a hospital robot due to its low processing power requirements and opportunity for integration with dynamic mapping.

Third, various methods for path planning and navigating static and dynamic environments were investigated. Global path planning methods are designed for static environment whereas local path planning methods are more focused on avoiding nearby obstacles. One of the main challenges found in local planning is trying to determine the best route to take when an obstacle is moving in the environment. Different methods used object tracking and trajectory prediction to attempt to avoid collisions but were unable to provide an optimal solution for multiple moving obstacles. One path planning method, however, demonstrated how to combine global and local planning into a single search-space divided into discrete states representing time windows. This solution allowed an A* search to find a viable path amongst moving obstacles in both space and time. This method, however, requires knowledge of all moving obstacles, visible or not, in a simulated environment, which is impossible to provide in a real-world environment. Some improvements could be made to reduce the overhead processing required for state-space updates in order to update the robot's path in real time.

Fourth, vibration-based floor classification methods were investigated as possible expanded sensing-capabilities of the envisioned prototype intended as part of this research programme. Detecting different types of floors can allow a robot to adjust its movements in pursuit of reducing noise levels and even detecting faulty floor-patches.

The investigated methods encountered in this literature review relied on the data acquired by a single 3-axis accelerometer that was either mounted on a "tail-like" probe or fixed directly onto the chassis. None of the reviewed methods attempted to classify different types of hard floor using omni wheels or Mecanum wheels. Omni wheels can be used in a holonomic configuration which, unlike a robot platform equipped with rubber wheels, enables various manoeuvres in tight spaces. This is despite causing significant vibration to the chassis, which must be addressed. Notwithstanding the need to address vibration, the absence of consideration of the relative strengths of a holonomic configuration appeared to be a gap in the literature which was ripe for investigation.

And finally, a broad overview of Human-Robot-Interaction (HRI) was presented, which included a taxonomy of the various roles involved in HRI. Safety is the foremost concern when introducing direct contact between humans and robots, both physical and social. This was evident in the literature, in that the focus was generally placed on safety and intention-recognition.

Whilst investigating various levels of interaction between humans and robots, including gaze-detection, it became apparent that new assistive robots should be capable of anticipating and preventing collisions with humans whilst maintaining a safe distance at all times. Intention-recognition, a central aspect of anticipating potential collisions, is often linked to human-robot collaboration work, but also allows for a more proactive form of helping in environments such as that which forms the operational context of this research.

Chapter 3

3 Mobile Robot Hardware Design

3.1 Introduction

One of the cornerstones of this thesis was to improve the capabilities of the current robotic platform in both sensing and mobility. The available research platform, codenamed EO3 within the Hamlyn Centre, was built on a Pioneer 3-DX [128] robot base, a non-holonomic differential wheel base with one laser range finder and one Kinect depth sensor.

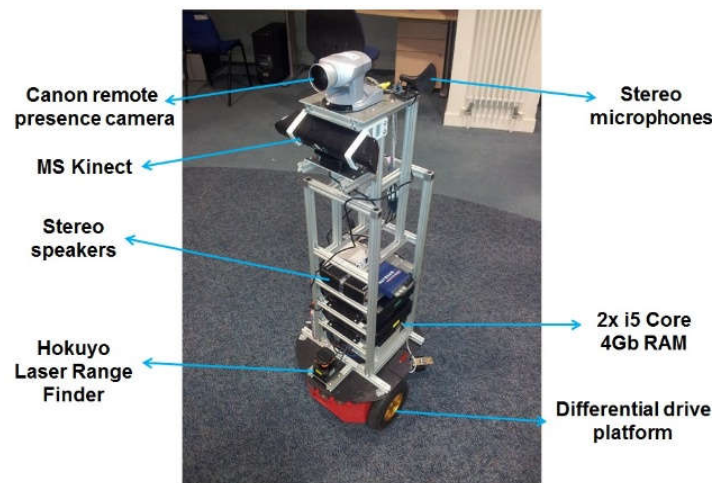


Figure 3.1 – Previous assistive robot prototype, codename EO3. This prototype was equipped with an off-the-shelf differential wheel base

The new design should incorporate the following:

- The base profile of the robot should be compact enough to allow it to move through commercial size doors and narrow passages without compromising its stability while turning. The narrowest space specification is shown Figure 3.2, and determines that the robot should have a turning radius of no wider than 60cm.
- The robot must be capable of holonomic movement to reduce the number of manoeuvres necessary to reach a certain position on the floor.

- The robot must also be able to travel at an average adult's walking speed to follow or guide people.

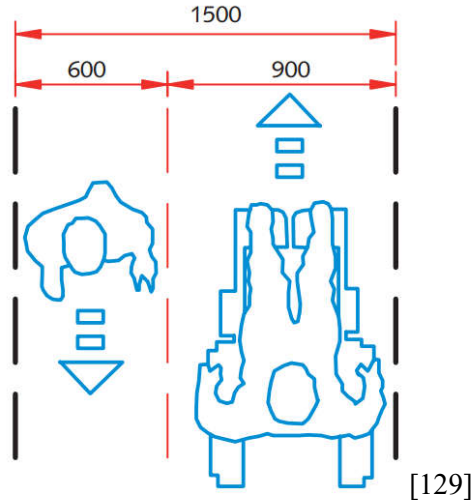


Figure 3.2 - NHS Hospital Building specification based on the Department of Health. The smallest corridor should allow for a wheelchair and a person to pass, where the robot would have to fit the same space as the person (600mm).

These physical requirements led to a 3-month research phase and development of different holonomic designs and their respective implementations that would satisfy the points listed above and prove to be both mechanically sound and relatively simple to implement.

3.2 Mechanical Design

The mechanical design stage of this project represented an interactive development of concepts of the holonomic base until the most cost-effective design was reached. Table 3.1 shows the four main wheel configurations considered, including an example photo to better illustrate how the 2D diagram translates as a real robot platform base.

The size requirement for the robot was to be able to fit within 60 cm diameter circle, which is based on the width of a commercial sized-door in the UK (80 cm). Hospital doors meant to be used by patients are up to 100 cm wide in order to allow easy wheelchair access, however, it is imperative that the robot's footprint is small enough to allow it to reach areas reserved for staff in order to perform deliveries and collections.


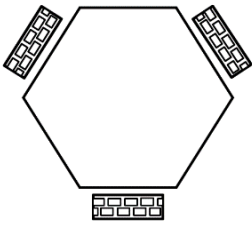

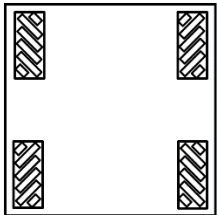

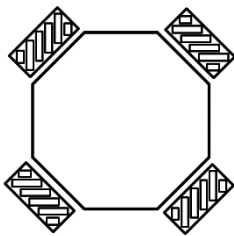

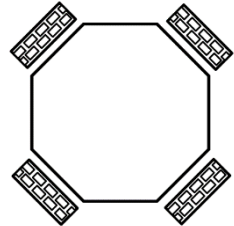
	Example	Diagram	Perceived Advantages	Actual Liabilities
(a)			<ul style="list-style-type: none"> • No movable parts necessary • Circular footprint 	<ul style="list-style-type: none"> • Reduced traction • Minimal internal space for components
(b)			<ul style="list-style-type: none"> • Easy to make • Optimises internal volume for components due to its rectilinear shape • Rectangular shape is ideal for Mecanum wheels 	<ul style="list-style-type: none"> • Square footprint increases minimum rotation radius • Requires suspension
(c)			<ul style="list-style-type: none"> • Circular footprint • Holonomic movement with cross configuration Mecanum wheels • Increased ground traction 	<ul style="list-style-type: none"> • Only uses two motors to move forward, making 50% of motors redundant • More vibration due to the reduced number of rollers in each wheel • Requires suspension
(d)			<ul style="list-style-type: none"> • Using four motors to move forward increases likelihood of achieving higher acceleration, thus improving overall control • Less vibration while moving 	<ul style="list-style-type: none"> • Vibration, while reduced, is still noticeable and affects camera tracking • Requires suspension

Table 3.1 – Comparison between types of holonomic base configuration considered during the design phase

The first base configuration considered was three omni directional wheels in an equilateral triangle arrangement (Table 3.1 (a)). This configuration has been used for robots like the home assistant robot prototype Armar-III [130]. The greatest advantage of this base is that it does not require any suspension to ensure all wheels touch the ground. With only three points of contact, the movement of each point is independent of the others, thus allowing it to overcome small irregularities on the floor while maintaining constant contact of all wheels. However, if the robot is pushed back, three wheels in a triangular configuration provide less traction due to the vectorial force decomposition. In addition, the hexagonal base design would provide less internal space for components compared to other solutions.

The next base configuration was the classical Mecanum base design (Table 3.1 (b)), which allows holonomic movement while maintaining a parallel wheel arrangement. This square base is also optimised for placing components with a cuboid shape, like batteries. The main disadvantage of using

four Mecanum wheels is that a suspension is required in order to maintain all four wheels in contact with the floor at all times. In an ideal scenario, four co-planar wheel axes would all touch a perfectly flat surface, but since the chassis can bend due to mechanical stress and not all surfaces are perfectly flat, a suspension is required to decouple two or more wheels in order to maintain contact. The suspension design and details are discussed in subtopic 3.3.4. The square perimeter of the base, when inscribed in a 60 cm circle, produces a much smaller internal space due to the corners of the square.

In order to overcome this size limitation, an octagonal shape was considered (Table 3.1 (c)). Adopting a cross-wheel configuration with Mecanum wheels has not been attempted before, but it represented potential new advantages to be explored. The base would work in the same way an omni-wheel-equipped base would, given only 45 degrees of rotation on its forward kinematics for each wheel.

This configuration was tested and worked as expected, however two drawbacks were discovered: First, the contact transition between the rollers created an excessive vibration on the base, which created more noise while the robot was running — an undesired effect for a service robot in a hospital. Second, the front of the robot is directed between the wheels of the robot, which forces the robot to move forward using only two diametrically opposite wheels. Using only two motors to move forward while facing the front would be energy inefficient as this is the main intended way of travel of the robot.

The final wheel configuration (Table 3.1 (d)) consisted of swapping the Mecanum wheels from the cross-wheel configuration base and using the omni wheels instead. The omni wheels available had two rows of rollers, which reduced the gap between roller transition and the overall vibration of the robot. In addition, if the robot is moving forward whilst facing forward, all four motors are used, which reduces the load for each motor.

3.3 Base Kinematics

The base kinematics are directly dependent on the type of wheels and how they are attached to the base. A representation omni wheel used in our assistive robot is shown in Figure 3.3.

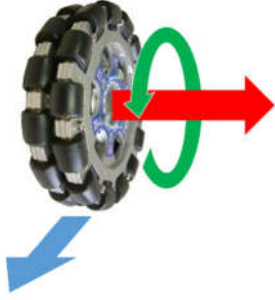
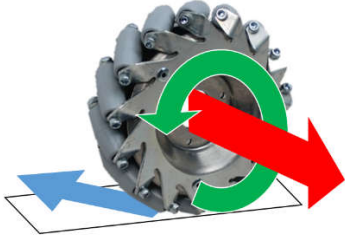
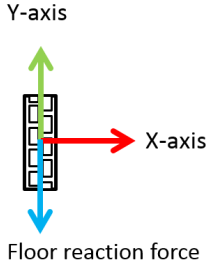
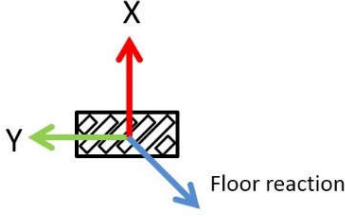
	Omni Wheel	Mecanum Wheel
3D Diagram		
2D Diagram		

Figure 3.3 – A comparison between omni wheel and Mecanum wheel regarding the floor contact force. Blue arrows represent the direction of the floor reaction for. Red arrows represent the wheel's rotation axis (X axis). Green arrows on the 2D diagram represent the Y-axis.

The red and green arrows represent the respective X and Y axes of the wheel's reference frame. The blue arrow indicates the direction where the wheel applies its ground velocity when it spins anti-clockwise around the X-axis (right-hand convention). The velocity vector is only present along the rotation axis of the bead touching the floor, hence the 45-degree direction offset on the Mecanum wheel diagram.

3.3.1 Forward and Inverse Kinematics

The forward kinematics of the base holds describes the conversion from the linear wheel speeds on the ground to the angular speeds used to control the motors. Figure 3.4 shows a diagram of the octagonal base design equipped with omni wheels (a) and Mecanum wheels (b). The position and orientation of each wheel's contact point is represented by the reference frame shown by green and red arrows. The blue arrows labelled W_n indicate which direction that respective wheel will contribute to the base's velocity when rotated forward.

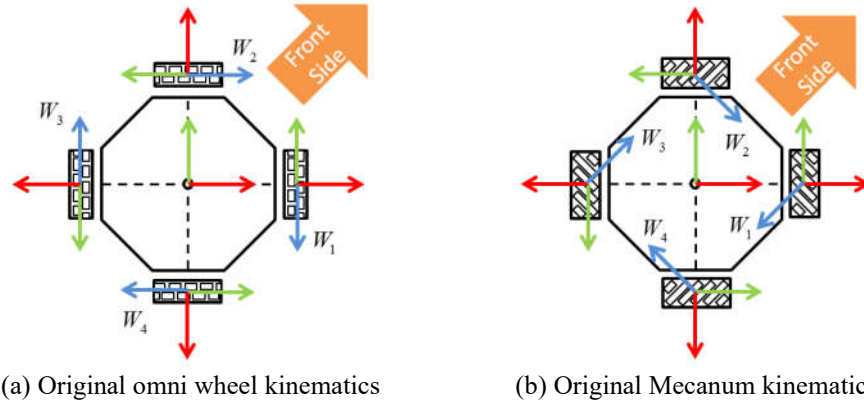


Figure 3.4 – Normal base orientation for the omni wheel (a), and the Mecanum wheel (b) versions of the base

The forward kinematics are composed of three individual equations that describe the linear and angular speeds of each of the wheels' contacts points.

$$Vx = 0.5W_2 - 0.5W_4 \quad (3.1)$$

$$Vy = -0.5W_1 + 0.5W_3 \quad (3.2)$$

$$\omega = -0.25 \frac{(W_1 + W_2 + W_3 + W_4)}{d} \quad (3.3)$$

Where Vx and Vy are the respective X and Y linear speed components and ω is the angular speed, the constant d is the distance between the contact point of the wheel on the ground and the centre of the base. Equations (3.1), (3.2) and (3.3) are then combined into the forward kinematics matrix shown in equation (3.4).

$$\begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} = \begin{bmatrix} 0 & .5 & 0 & -.5 \\ -.5 & 0 & .5 & 0 \\ -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \quad (3.4)$$

The Mecanum wheel forward kinematics matrix is also derived from the similar set of equations (3.5), (3.6) and (3.7).

$$Vx = -0.25W_1 + 0.25W_2 + 0.25W_3 - 0.25W_4 \quad (3.5)$$

$$Vy = -0.25W_1 - 0.25W_2 + 0.25W_3 + 0.25W_4 \quad (3.6)$$

$$\omega = -0.25 \frac{(W_1 + W_2 + W_3 + W_4)}{d} \quad (3.7)$$

The final matrix can be seen in equation (3.8).

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} -.25 & .25 & .25 & -.25 \\ -.25 & -.25 & .25 & .25 \\ 1 & 1 & 1 & 1 \\ -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \quad (3.8)$$

The last step is to rotate the initial orientation of the base so the front side of the base is aligned with the X-axis. The result of rotating the base can be seen for both wheels in Figure 3.5.

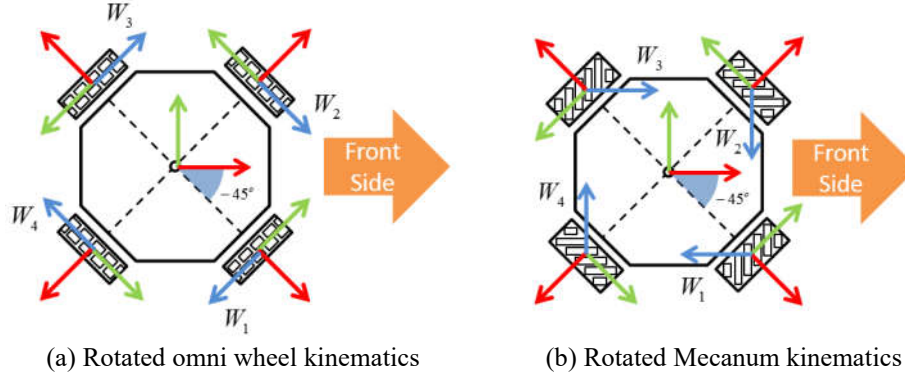


Figure 3.5 – Rotated base orientation for the omni wheels (a), and the Mecanum wheels (b) versions of the base. When the frames of reference are rotated, they align with the robot's X axis.

The modified forward kinematic equations for the omni and Mecanum wheel can be seen in equations (3.9) and (3.10), respectively.

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(-45) & -\sin(-45) & 0 \\ \sin(-45) & \cos(-45) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & .5 & 0 & -.5 \\ -.5 & 0 & .5 & 0 \\ 1 & 1 & 1 & 1 \\ -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \quad (3.9)$$

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(-45) & -\sin(-45) & 0 \\ \sin(-45) & \cos(-45) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -.25 & .25 & .25 & -.25 \\ -.25 & -.25 & .25 & .25 \\ 1 & 1 & 1 & 1 \\ -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} & -\frac{1}{4d} \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \quad (3.10)$$

The value of d is 0.291 metres, which simplifies equations (3.9) and (3.10) into equations (3.11) and (3.12).

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} 0 & -0.3536 & 0 & 0.3536 \\ 0.3536 & 0 & -0.3536 & 0 \\ 0.8591 & 0.8591 & 0.8591 & 0.8591 \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \quad (3.11)$$

$$\begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.3536 & 0.3536 & -0.3536 & -0.3536 \\ 0.8591 & 0.8591 & 0.8591 & 0.8591 \end{bmatrix} \times \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \quad (3.12)$$

3.3.2 Inverse Kinematics

The inverse kinematics provides the exact reverse operation of the forward matrix by converting the linear and angular velocities into linear wheel speeds. Equations (3.11) and (3.12) are inverted using the Moore–Penrose pseudoinverse. The final inverse kinematic for the omni wheel can be seen in equation (3.13).

$$\begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} = \begin{bmatrix} 0.707 & 0.707 & 0.291 \\ -0.707 & 0.707 & 0.291 \\ -0.707 & -0.707 & 0.291 \\ 0.707 & -0.707 & 0.291 \end{bmatrix} \times \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \quad (3.13)$$

And the inverse of the Mecanum wheel type can be seen in equation (3.14).

$$\begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} = \begin{bmatrix} 0 & 1.414 & 0.291 \\ -1.414 & 0 & 0.291 \\ 0 & -1.414 & 0.291 \\ 1.414 & 0 & 0.291 \end{bmatrix} \times \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \quad (3.14)$$

3.3.3 Base Velocity Close Loop Control

The kinematics equations are employed in the close loop control of the base velocity. The control of the robot's velocity utilises a motion planner node that dictates that current target velocity to the base controller node which converts the base velocity vector to the wheel speed vector using the inverse kinematics equation. The base controller node is also responsible for the PID control of each motor controller. It sends the intended motor speed to the desired motor controller, and in return receives the change in encoder counts on that motor periodically. The base controller node adjusts the amount of power used to keep each wheel at the desired speed and also reports the current velocity of the base back to the motion planner. The motion planner uses the velocity vector provided by the calculated robot's path to generate individual wheel speeds using the forward kinematics equation. A diagram with the entire speed control pipeline is shown in Figure 3.6

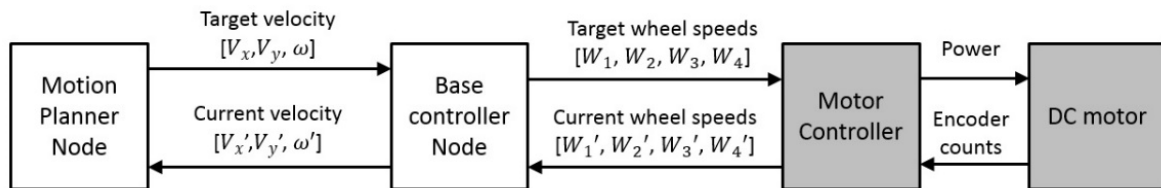


Figure 3.6 – Base velocity close loop control with software (in white) and hardware (in grey) elements.

3.3.4 Prototypes and Suspension Test

Small indentations and saliences on the surfaces combined with the small bending on the chassis caused by mechanical stress can cause one wheel to leave the ground, thus compromising control of the robot. Rubber wheels do not suffer from this effect on planar surfaces as they are able to deform and compensate for height variations. On the other hand, hard material wheels such as the Mecanum and omni wheels need a suspension to adapt their clearance to the ground in order to maintain full contact.

Three different ideas were tested to verify the viability, relative merits and drawbacks of each type of suspension. First, a no-suspension scenario was used as a control. Four Dynamixel MX-28 motors were connected together in a rigid frame configuration and four small-scale 3D printed Mecanum wheels were used to test the control of the base. Despite the reduced scale of the test, it was noticeable that the model could not be steered towards the desired direction due to slippage and lack of wheel contact. Different surfaces were tested, including carpet and wood, and extra weight was added to increase the friction between the surface and the wheel, however, similar results were observed.

The second possible solution was to use an independent suspension design. This solution is very common in the automotive industry for its high effectiveness and quick response to different types of terrain, but it comes with a great increase in the overall design complexity of the system. A LEGO prototype was constructed in order to test the independent suspension concept and was tested on the same surfaces. The second prototype demonstrated more control than the first prototype with no suspension, and it was even able to move over small obstacles such as thin magazines without any significant change in its course.

The last prototype consisted of a 3D-printed scaled version, with which the wheels of the robot would be inside the 60 cm diameter circle intended as the robot's maximum footprint. The prototype had a rotation axis around its centre, allowing half of the robot to incline while moving over an indentation or bump on the ground. Compared to the second prototype with independent suspension, the swivel suspension performed just as well on the same surfaces. One noticeable drawback, however, is that the main body of the robot would have to be fixed to one-half of the base. If a wheel from the

half where the main body is attached to, rolled over a bump on the surface, the entire body of the robot would incline in the opposite direction.


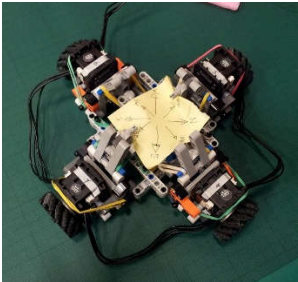

	Example	Perceived Advantages	Actual Liabilities
Rigid		<ul style="list-style-type: none"> • Simplest construction, as it doesn't use a suspension 	<ul style="list-style-type: none"> • Prototype demonstrated impaired control of movement due to momentary loss of contact with surface
Independent		<ul style="list-style-type: none"> • Highly adaptable to irregular surfaces with small height variation • Capable of maintaining vertical axis near perpendicular to the ground when overcoming small obstacles 	<ul style="list-style-type: none"> • Very complex to design and manufacture an independent suspension system from scratch • Internal space compromised to allocate suspension arms
Swivel		<ul style="list-style-type: none"> • Considerably simpler to build compared to the independent suspension 	<ul style="list-style-type: none"> • The main body of the robot has to be fixed to one half of the suspension, which causes it to tilt when that half is overcoming an obstacle

Table 3.2 – Summary table comparing different types of suspension tested

After comparing all three prototypes, it was decided to adopt the swivel suspension in the final base design. The extra mechanical complexity of the independent suspension system greatly outweighed the advantages of maintaining a somewhat more perpendicular position of the robot's body compared to the ground plane when moving over small bumps and indentations. **Error! Reference source not found.** summarises the advantages and disadvantages of the three prototypes previously discussed.

3.4 CAD Designs of the Chassis

Once the type of suspension was selected, the next development stage was the 3D CAD design of the robot's chassis. Maximum mass, speed and acceleration were taken into account when selecting the

motors and the shape of the frame for the base. The selected motors have a 30-watt peak power, with 192 RPM maximum output shaft speed and 1.4 Nm rated torque.

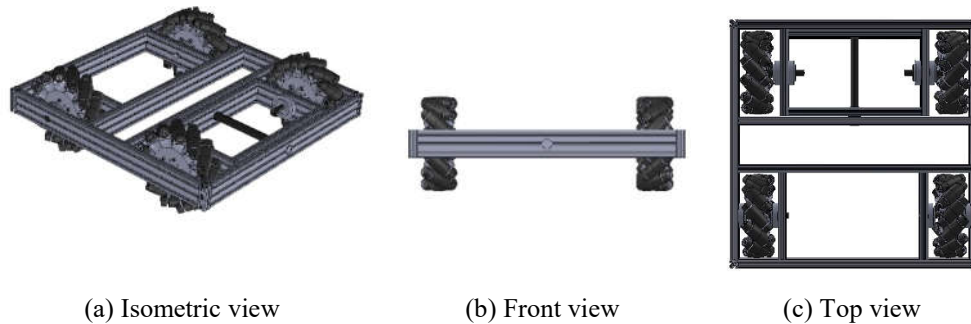


Figure 3.7 – First suspension design concept. The base can be seen from an isometric view (a), a front view (b) and a top view (c)

The first base design was inspired by the Mecanum base seen in Figure 3.7 (a). It used 8” low vibration Mecanum wheels and had a square ground. However, in order to accommodate these large wheels and still maintain stability, the size the frame would be larger than the size requirements. This solution presented maximum internal space for housing electronic components, and the square metal frame around the base provided extra rigidity at the cost of increasing the clearing radius of the turns. The design had to be changed to fit a 50x50 cm-limit size, which created problems when trying to stack components vertically and removing the external metal square frame.

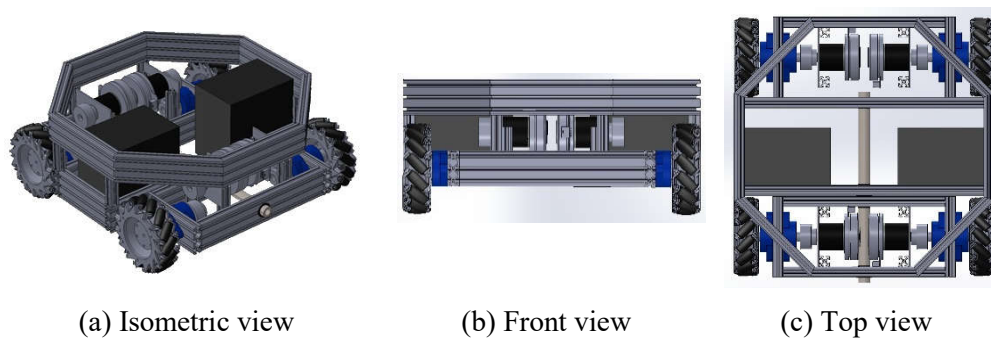


Figure 3.8 – Second suspension design concept. The base can be seen from an isometric view (a), a front view (b) and a top view (c)

The second base design, Figure 3.8, used a new set of Brushless DC motors for its small volume to power ratio and allowed the addition of electrical brakes on each wheel’s axis. The brakes were intended to prevent the robot being pushed back if physically interacting with a human, but they were deemed not essential for the first prototype of the robot. Further versions of this robot will aid people standing

up from chairs or beds, and electrical brakes will be more useful in these situations. The top section of the design was reduced to an octagon for both aesthetic purposes and reducing the overall volume of the robot. Despite the more compact design, it was decided that the ideal robot footprint would be circular, which introduced an extra degree of challenge to the mechanical design.

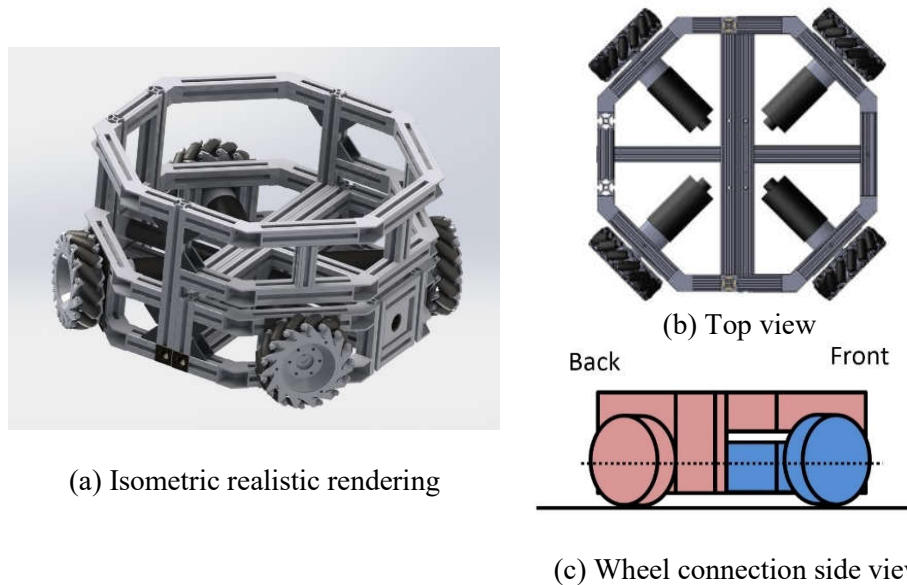


Figure 3.9 – Third and final design. Pre-rendered view (a) along with the top view (b) and side view (c) showing the two parts of the suspension in pink and blue.

The third and final design, Figure 3.9, was based on a complete octagonal profile as it better resembles a circle while still maintaining opposite sides for the perpendicular axis configuration. The swivel suspension was created by introducing a 20 mm diameter, 2 mm thick steel tube through the centre of the base.

Figure 3.9 (a) shows a realistic rendering of the base used as the final design for the project committee while Figure 3.9 (b) shows top view of the robot base. The symmetry of the design also contributed to keep the centre of gravity of the robot at the geometric centre of the base. The design used a symmetric passive suspension, where the top half of the structure is fixed to the rear of the structure. Figure 3.9 (c) shows a colour-coded side-view diagram of the robot base. The red section represents the back structure of the base along with the connected back wheels. The front structure is shown in blue along with the connected front wheels. The back and front wheels can move independently around the central axis created by the steel tube.

3.5 Suspension test

Once the base frame was completed and all necessary components to control it were added, a series of tests took place to evaluate how the suspension would perform under various conditions.

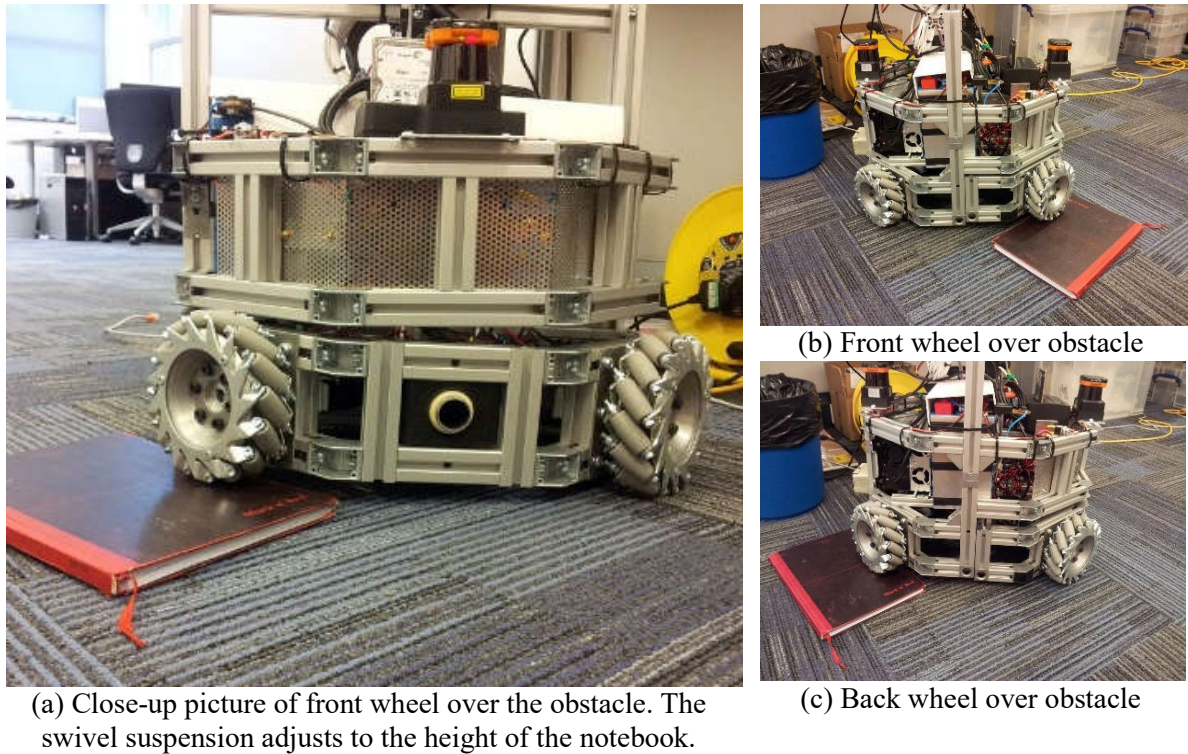


Figure 3.10 – Demonstration on how the swivel suspension works. Notice that the front wheels rotate around the centre axis, all four wheels maintaining contact on the floor and obstacle

Figure 3.10 shows images of the passive swivel suspension in action on a hardcover notebook. On a fully flat terrain the omni-directional nature of the suspension frame creates the desired situation in which all four wheels touch the floor. If any of the four wheels momentarily stops making contact with the floor whilst the robot is in transit, however, that wheel's speed-contribution to the overall velocity vector would be nullified, thus altering unhelpfully the robot's direction and speed of travel. It stands to reason that in terrain in which an obstacle is encountered, as much as is reasonably possible this phenomenon should be avoided.

The nature of the obstacles likely to be encountered were considered to be limited and of negligible risk of toppling the robot. Therefore an inherent compromise was accepted, in context of the function of the suspension frame and the omni-directional wheels. In conditions in which the robot encounters,

for example, a floor change whilst traveling in a forward direction (in the simulation above this was depicted by means of a notebook), the front wheel(s) is/are raised but the base remains parallel to the floor. This ensures that the robot remains plumb at ninety degrees to each varying surface, whilst also ensuring that all four wheels are making full contact.

In the instance of an upward or downward inclining ramp it was hypothesised that with sufficient distance between the chassis and the floor, the robot would use the built-in IMU unit to recalibrate the sensors in context of the ramp's degree of inclination. It was accepted that obstacles were unlikely to be encountered firstly on the back wheels so the inherent limitations of the chassis structure were permitted to remain as-is, in that they do not result in the base remaining parallel to the floor. This is because the back wheels are affixed to the robot's chassis, whereas the front wheels were designed to accommodate slight changes in floor plane and the construction of the chassis permits the front wheel to pivot by means of the steel tube axle. This can be seen diagrammatically in Figure 3.9 (c).

3.6 Electrical and Electronic Design

3.6.1 Power Grid system

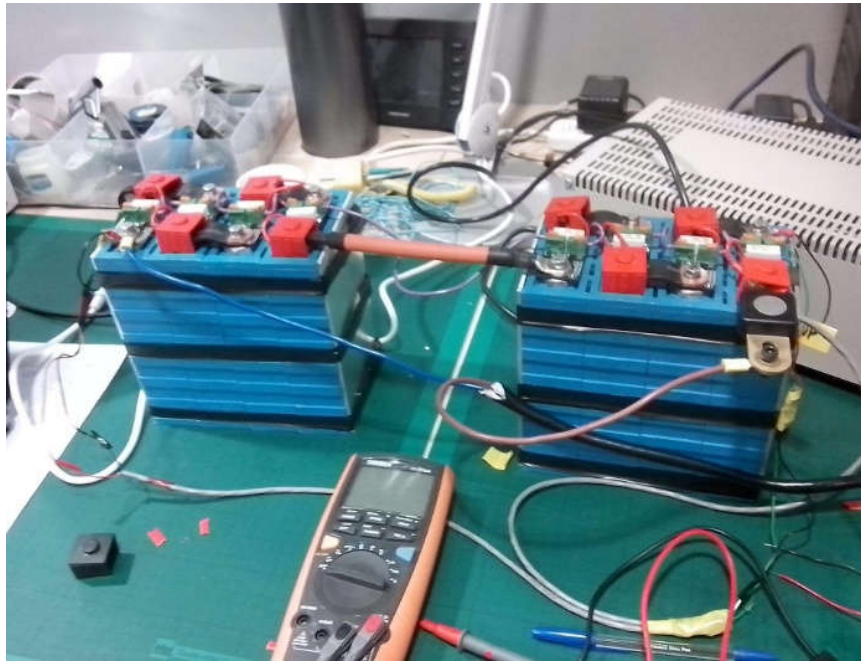
The power system of the robot was designed to provide both versatility when using it on mobile and stationary experiments, along with greater autonomy compared to its predecessor. Figure 3.11 shows a simplified diagram of the new robot's power system with all its major components and connections. One of the main differences between the new and old design is the transition between 12V to 24V as the primary power source voltage. At 24V, drive motors are significantly more efficient and achieve higher speeds compared to their counterparts running in 12V. In addition, future servo actuators, e.g. for a robot arm, run predominately at 24V DC.

the robot had to operate for more than 1 hour, as it would run out of battery. The new design, seen in Figure 3.11, addressed these limitations by carrying both an on-board high-power battery charger and a high-power supply unit. Recharging the batteries could now be done by connecting the base AC input socket to the mains socket via a normal power cord. When the robot is connected to 240V AC, the power supply and the battery charger are turned on and the batteries start charging automatically. At the same time, a relay disengages the batteries from the rest of the power grid and engages the power supply unit, thus providing power for the robot without draining the batteries. This feature is specifically useful when the robot is required to be online for extended periods of time for debugging without needing to move. The internal power supply was selected to be powerful enough to keep the robot running indefinitely while stationary, while the batteries can provide running autonomy for up to four hours under normal conditions.

The last addition to the new robot's power system was its hardware-based emergency stop button. Previously, the robot's motors were stopped via the software embedded into its Pioneer base. The new emergency stop system (Figure 3.11) cuts the power of the motors via an emergency relay controlled by the emergency button. Once the button is pressed, it latches into an open state, thus disengaging the power into the motor controllers. Once the button is reset, the relay is re-engaged and motors are re-activated. This is a safer solution because it is independent from software error.

3.6.2 Batteries

The batteries used in the new robot design were supplied by Green Motor Sports Ltd, a battery supplier for electric car manufacturers in the UK. The Li-Ion batteries supplied have twice the energy density of a normal lead-acid battery commonly found in cars with internal combustion engines. These batteries can provide a 24V DC output with a constant 48Ah power output. A lead-acid battery with the same power output would weigh twice as much and would require 8–12 hours to charge from empty, compared to the quick two hours required by these batteries. One of the main advantages of this battery management system is that an entire profile of the battery usage can be visualised in real time with an external laptop through a USB connection.



(a) Close-up picture of front wheel over the obstacle



(b) One pack of four cells provides 12V DC



(b) Individual cell management unit

Figure 3.12 – Pictures of the battery management system and its components in detail.

The BMS control module is responsible for adjusting the voltage of each battery cell while charging, by using a shunt resistor on the control PCB of each board. Figure 3.13 shows a picture of the user interface of the software provided with the BMS module that reads individual battery cell voltage levels and temperature.

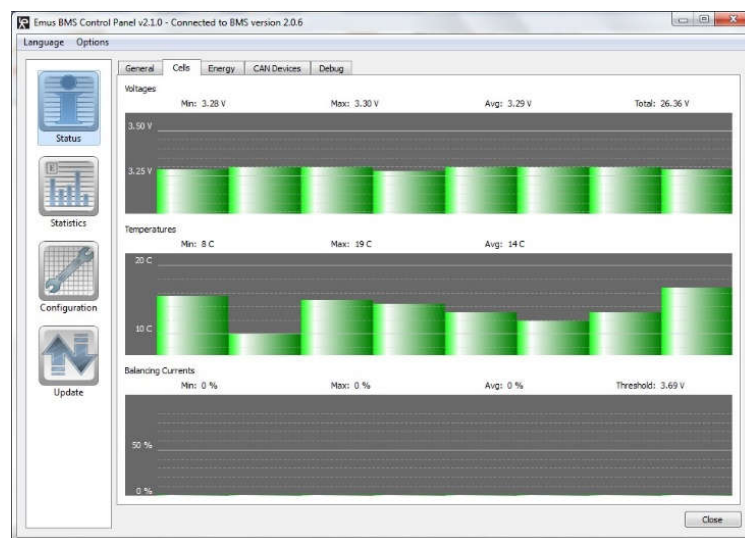


Figure 3.13 – Voltage and temperature levels of individual battery cells through the BMS

The BMS interface is used to log battery usage, control internal temperature levels, set individual cell voltages, as well as more specific functions regarding overall maintenance of the cells.

3.6.3 Sensors

The type of sensors and their location on the robot's body play a fundamental role in determining the robot's degree of accuracy in perceiving the world and its moving obstacles. The previous robot prototype had one single Lidar, which provided 270 degrees of coverage from the robot's centre up to 30 metres away from it. The new design platform received two identical Lidars placed on diametrically opposite sides of the robot's base in order to achieve 360 degrees of coverage. Figure 3.14 shows a diagram with the major navigation and reconnaissance sensors.

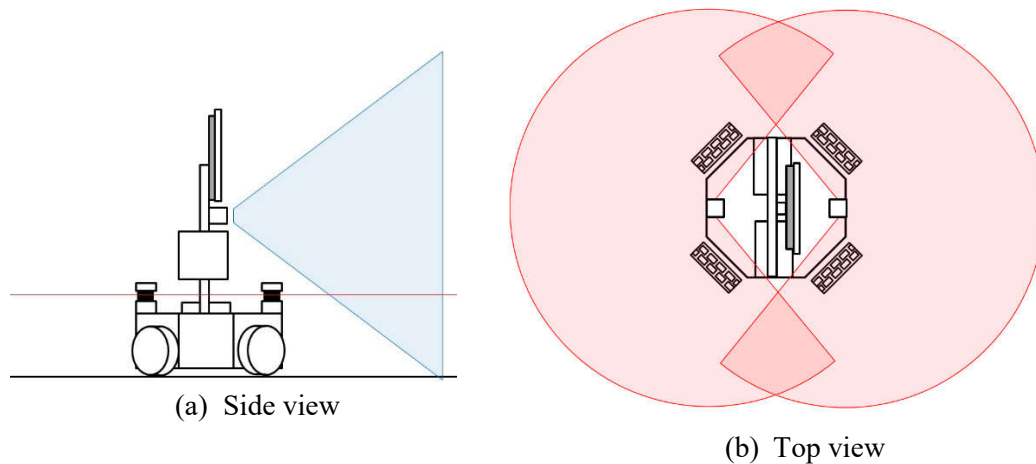


Figure 3.14 – Sensor positions and sensing regions on the robot. The side view shows the depth camera's field of view and the Lidar's scanning plane parallel to the ground. The top view shows the angle coverage of both Lidars, although with their range radius reduced to fit the diagram.

In Image (a), a side view of the robot is depicted, with a horizontal red line representing the laser range finder and a blue trapezoid representing the Kinect's vertical field of view. The top view, in Image (b), shows the coverage of both Lidar sensors. The addition of multiple depth sensors was also considered but later declined, as this would increase the amount of real-time data to process and consequently decrease the amount of time left for core computations. The current choice of sensors allows the robot to sense obstacles and rotate when necessary to face the depth sensor to analyse any object in more detail.

3.7 Conclusions

This chapter described the major milestones in the design process of the new assistive robot platform. The design started with the initial specifications based on the working environment in which the robot was intended to be used. Different holonomic driving mechanisms were compared and the cross-wheel configuration on an octagonal base frame was selected as the best compromise between internal space and flexibility to try different types of wheels. Next, three solutions for suspension were tested with the aid of prototypes, and the swivel suspension was demonstrated as the most cost-effective solution for maintaining four wheels on holonomic drive on a flat surface at all times. The major CAD design ideas were discussed, followed by the electrical and electronic design stages. Finally, a suspension test was conducted showing how the robot was able to overcome small obstacles on the ground up to 20 mm high.

The final robot platform can be seen in Figure 3.15 equipped with omni wheels. The decision to use omni wheels over the Mecanum wheels at the end of the hardware design phase was based on two factors:

1. **Vibration and noise levels:** The Mecanum wheels have 15 rollers each compared to the 28 rollers arranged in two rows of 14 rollers on each omni wheel. Both wheels have overlapping rollers but the extra number of rollers in the omni wheels end up reducing the gap between the rollers thus creating a smoother motion. The Mecanum wheels produced more sound while rolling compared to the omni-wheels; the difference was noticeable to any bystanders.
2. **Motor control when moving forward:** Due to the physical orientation of the robot's front side relative to its base, only two motors are required in order to move the robot forward and backward when the Mecanum wheels are used; four are required when omni wheels are used. Moving in diagonals inverts the number of motors. The problem arises when the robot is required to move forward for the majority of its travel distance. In this case, only two motors would be used, thus compromising the robot's top acceleration and maximum speed.

A comparison table of the robot's physical characteristics compared to the state of the art can be seen in Figure 3.15 below.




			
Category	RP-Vita	New Assistive Robot	Pepper
Height	168 cm	121cm	120 cm
Footprint	58cm x 58cm	60cm x 60cm	42.5cm x 48.5cm
Weight	79.4kg	58.2 kg	28kg
Top Speed	1.5 ms ⁻¹	1.2 ms ⁻¹ (limited)	0.89 ms ⁻¹
Battery Autonomy	4-5 hrs	4-6 hrs	12 hrs

Figure 3.15 – Comparison table between the physical characteristics of the proposed New Assistive Robot and two of the latest assistive robots used in hospital environments

What can be noted from Figure 3.15 above are a number of matters of interest. Firstly, the new assistive robot is mid-range in nearly all of the categories, save for footprint of the prototype. Nonetheless, even with this footprint the prototype meets the recommendations presented in a document published by the Department of Health [131] as can be seen in Figure 3.2. Next, in terms of battery autonomy, the new assistive robot exceeds the RP-Vita prototype by 1 hour, depending on its in-situ deployment and range. And finally, whilst the Pepper is a comparatively light-weight assembly, the new assistive robot compares favourably, even as a prototype, to the RP-Vita. It can be seen that even before optimising choice of materials, it is 27% lighter than the RP-Vita as a brought-to-market product. This is in context of the fact that the new assistive robot is only (software-limited) 20% slower at top speed than the market-ready RP-Vita. It is anticipated that materials optimisation would improve both weight and, in turn, top speed.

Chapter 4

4 Floor classification

4.1 Introduction

New research groups and companies such as InTouchHealth [6] have already started introducing new tele-operated robotic assistants to hospitals and office workplaces as the need for a new automated workforce grows. These new robots often have a holonomic drive in order to minimise their movement footprint when moving through crowds, so as to reduce the disruption to people in the environment.

Robots designed to operate in outdoor environments are usually equipped with large rubber wheels to allow them to cross irregular terrain, whereas robots designed for indoor applications are equipped with smaller plastic wheels and have no suspension.

When a robot equipped with omni wheels or Mecanum wheels is moving, a noticeable vibration comes from the impacts between the edge of the rollers and the floor while the wheels are spinning. This vibration can create loud noises on hard surfaces, which can then be minimised by reducing the speed of the robot. On soft surfaces such as carpets, the rollers tend to “sink” a few millimetres on the carpet, thus creating extra drag and reducing the robot’s overall speed and performance. If the robot is able to identify what type of surface it is moving on, it could re-adjust its speed and path planning parameters to better cope with different surface conditions.

The literature on identifying different floor types presented in Chapter 2 has demonstrated that classifying a vibration signal requires the least expensive hardware and processing power overall. This led to the addition of a single IMU unit directly attached to the chassis of the assistive robot, as shown in Figure 4.1.

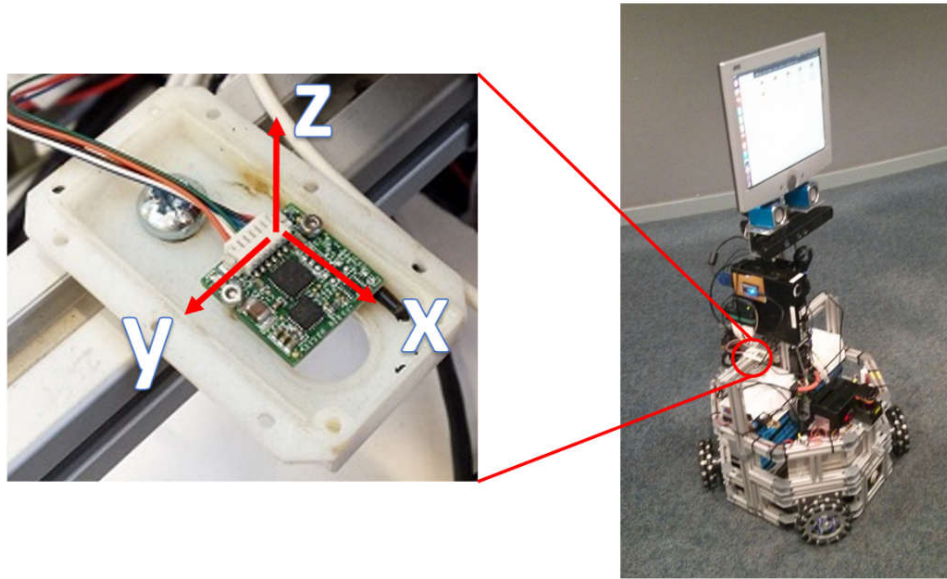


Figure 4.1 – IMU unit in detail and its location on the robot's chassis. The Z arrow on the IMU indicates the vertical direction of the vibration.

Unlike previous floor classification research, which relies on rubber tires [83-85], the use of omni wheels introduces significant noise to the vibration signal as the speed of the robot increases. Since the nature of the noise itself depends on too many variables, pre-processing the noisy signal would most likely remove some of the features from the vibration signal itself. Instead, the raw signal is used to directly generate all the features to be used by the classifier. The complete data processing pipeline can be visualised in the diagram in Figure 4.2.

First, the raw signal is recorded by the IMU, which is split into overlapping 1024-sample sliding windows. Next, two sets of features (statistical and Power Spectrum Density (PSD)) are extracted from each window. After that, both sets of features are concatenated into a single 523 feature set, which is then simplified using Principal Component Analysis into 10 components. Finally, the 10 components are normalised and used as training parameters for the Extreme Learning machine neural network classifier.

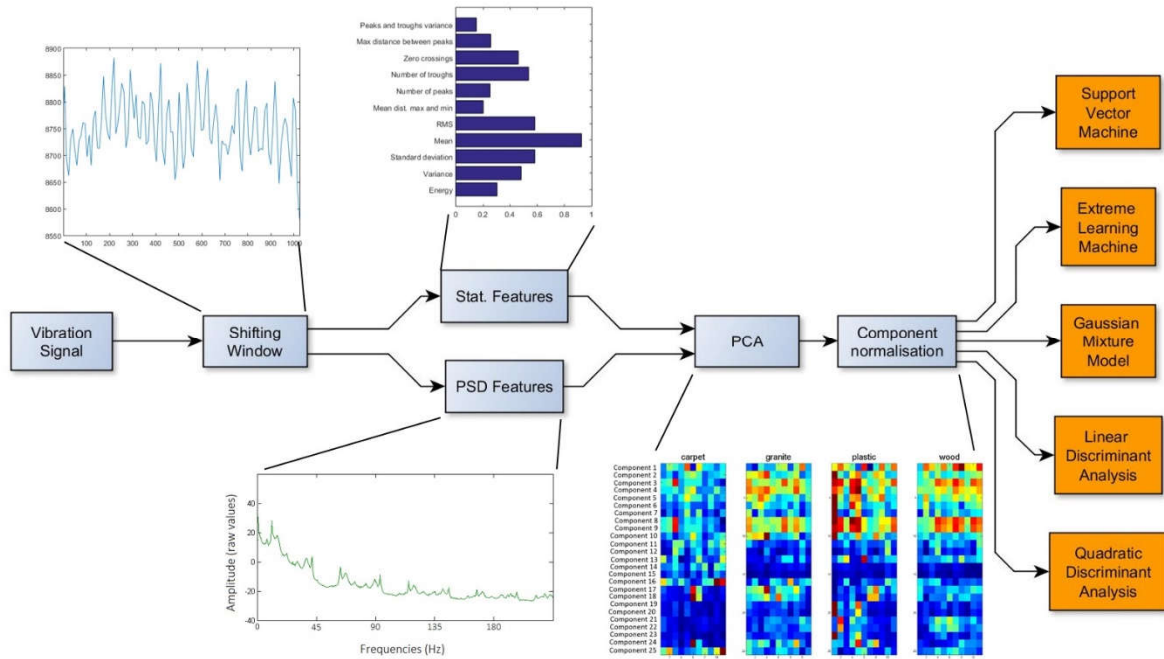


Figure 4.2 – Complete data processing and classification pipeline. The features extracted from the sample in the shifting window are reduced and normalised before being used for training. Visualisations of the data in each stage are shown to illustrate the process, but their meaning is covered in detail in the subsequent sections.

4.2 Data collection

The sensor selected for detecting the ground vibration was the accelerometer inside a 9-axis 3.3V IMU unit set to a 446 Hz sampling rate determined by the maximum bandwidth between the microcontroller board and the computer. The IMU unit was connected to an mBed NXP LPC1768 development board which was programmed to read the raw data and transmit it via a serial link to the main robot’s computer.

The signal in Figure 4.3 depicts 72 seconds of vibration signal recording, where the robot is performing a variety of manoeuvres, including moving in a straight line and rotating. The respective recorded vibration for each of the actions is clearly labelled on the figure. The sensor was positioned parallel to the ground on the chassis in order to reduce any cross-talk between axes during recording. The measurement of the Z-axis of the sensor corresponds to the full extent of the chassis.

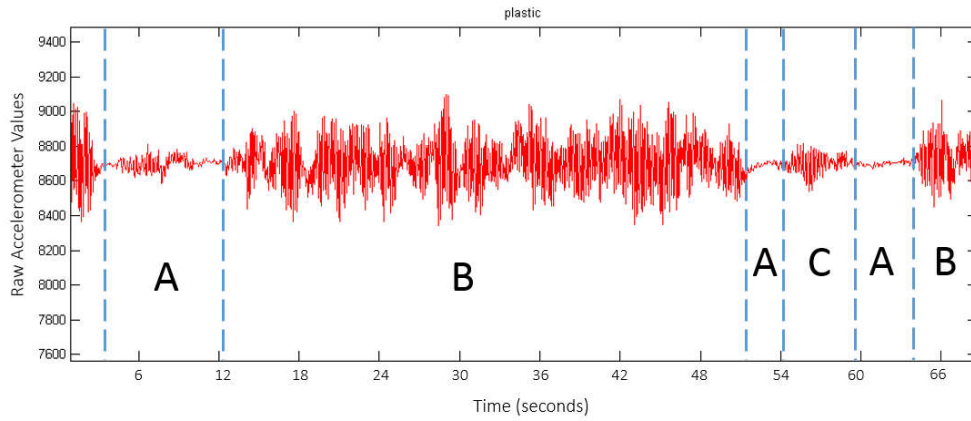


Figure 4.3 – Example of vibration recorded by the IMU during recording stages: (A) robot not moving; (B) robot moving in one direction; and (C) robot rotating

When considering that both the type of the floor and the linear and angular speeds of the robot can affect the vibration signal, a series of manoeuvres with different speeds were designed to capture a wide range of possible movements the robot may perform at any time.

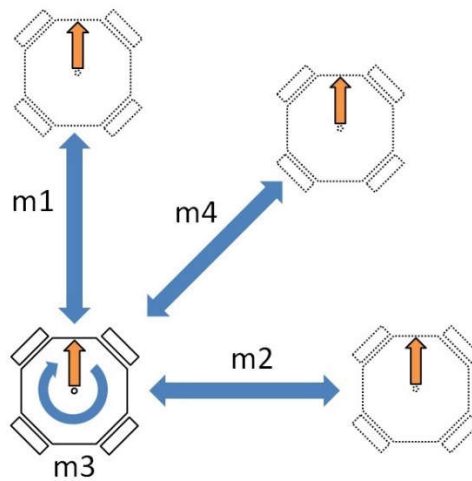


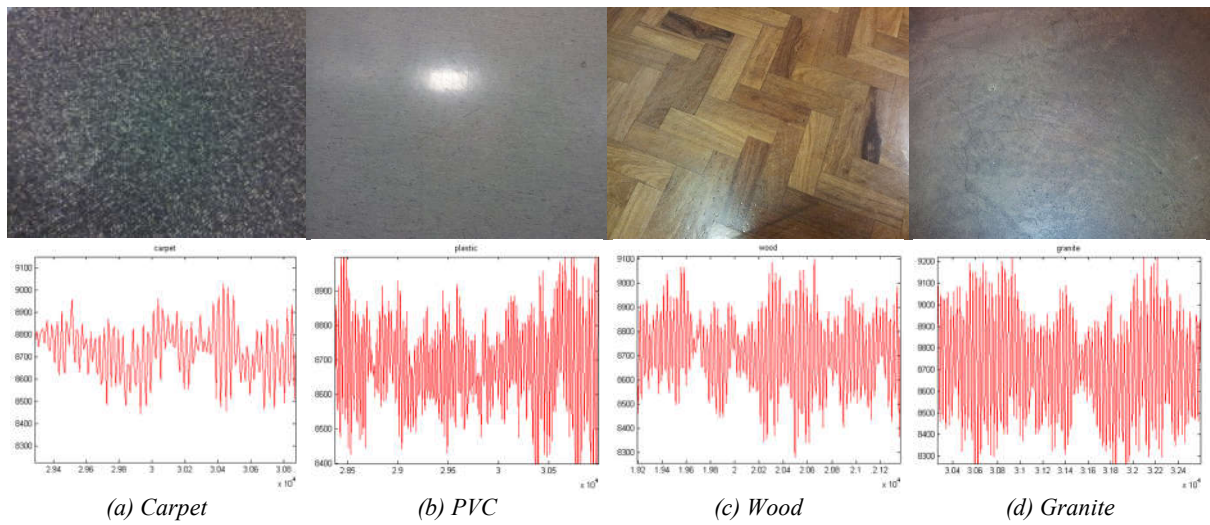
Figure 4.4 – All four types of movement were executed on each floor to ensure coverage of all degrees of freedom of the base. Orange arrows indicate the front side of the robot

Movements m1, m2 and m4 consisted of moving the robot for 10 seconds back and forth on vertical, horizontal and diagonal directions, respectively, at speeds from 0.2 to 1.0 ms^{-1} spaced at 0.1 ms^{-1} . These movements were selected to explore the effects of vibration on linear movement.

Movement m3 consisted of rotating the robot clockwise for 10 seconds at 0.6 , 0.9 , 1.2 and 1.5 rads^{-1} . This movement was selected to explore the effects of vibration on angular movement. Due to space

constraints, combining both angular and linear movement at different speeds was not possible, as it would drive the robot in larger and larger circles.

Four different surfaces were selected for the data collection: carpet, PVC (plastic), wood and granite, as shown in Figure 4.5, along with a four-second sample signal of their respective measured vibrations. These four floors are commonly found in both offices and hospital environments. As such, they represent a wide range of hardness, ranging from soft to hard, better illustrating the overall accuracy of the methods presented.



4.3.1 Statistical Features

Several techniques for classifying different surfaces using vibration recorded by an IMU were discussed in the literature survey in Chapter 2, each with their own set of statistical features selected to yield the highest accuracy during classification. The list below shows the features that provided the best differentiation amongst different types of floor across all reviewed publications.

1. Energy
2. Variance
3. Standard deviation
4. Mean
5. RMS
6. Mean distance between max and min values
7. Number of peaks
8. Number of dips
9. Zero crossings
10. Maximum distance between peaks
11. Peaks and troughs variance

Different combinations of statistical features were tested, leaving these 11 features as the optimal combination for differentiating the vibration signal recorded while the robot was moving on different surfaces. However, the overall accuracy using only statistical features was around 83%, due to the low SNR introduced by the wheel rollers touching the floor during the robot's movement.

4.3.2 Frequency-based Features

The Power Spectrum Density (PSD) features are frequency-based features introduced in an attempt to improve the overall accuracy of the classification. The PSD features demonstrated that more features could be detected from the floor type almost independently from how fast the wheels were spinning. The graph in Figure 4.6 shows how the amplitude of the PSD features changes depending on the robot's base velocity, and by extension, the wheel's velocity. The PSD features were extracted using a 1024-points FFT resulting in a 446-feature vector equally distributed between frequencies from 0 to 223Hz.

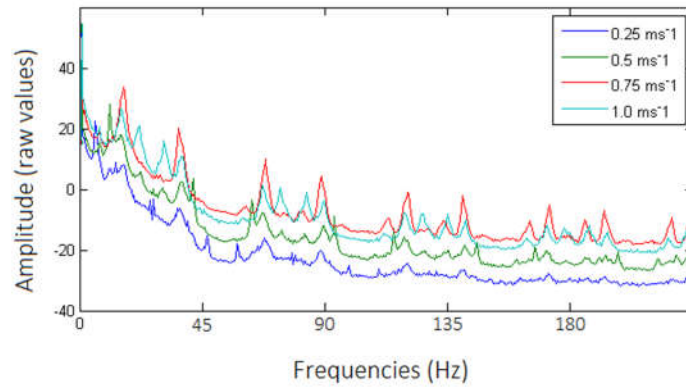


Figure 4.6 – Power Spectrum Density of signal recorded while the robot performed the same movement at different speeds - shown in different colours

From 0.25 ms^{-1} to 0.75 ms^{-1} , there is a steady increase on the overall amplitude of the signal. As the wheels spin faster, the point of contact on the roller's edge hits the ground with higher speed, thus generating more vibration. When the robot is moving at 1.0 ms^{-1} , however, there is a slight decrease in the overall signal amplitude. This is explained by the slipping of the wheels due to the high-speed rotation. As the wheels rotate faster, some edges of the rollers miss contact with the floor, and this phenomenon results in an overall “smoother ride”. For a robot moving within a caring centre or hospital, 1.0 ms^{-1} is too fast for safe operation, hence the nonlinearities of omni wheels slipping and skidding are not a concern for this scenario.

4.3.3 Final PCA Feature Set

The final phase of the feature extraction stage is combining both statistical and frequency-based features into one single feature vector. Both statistical and frequency features combined added up to 523 (11 statistical and 512 PSD frequency features); hence, dimensionally, reduction was necessary to speed up the training phase. Principal Component Analysis (PCA) and Kernel PCA were tested to evaluate which represented the data more accurately. PCA was selected as it was able to represent data with 95.81% of its original variance using only the first 10 components and required much less time to process during training and fitting, which is depicted in Figure 4.7 below.

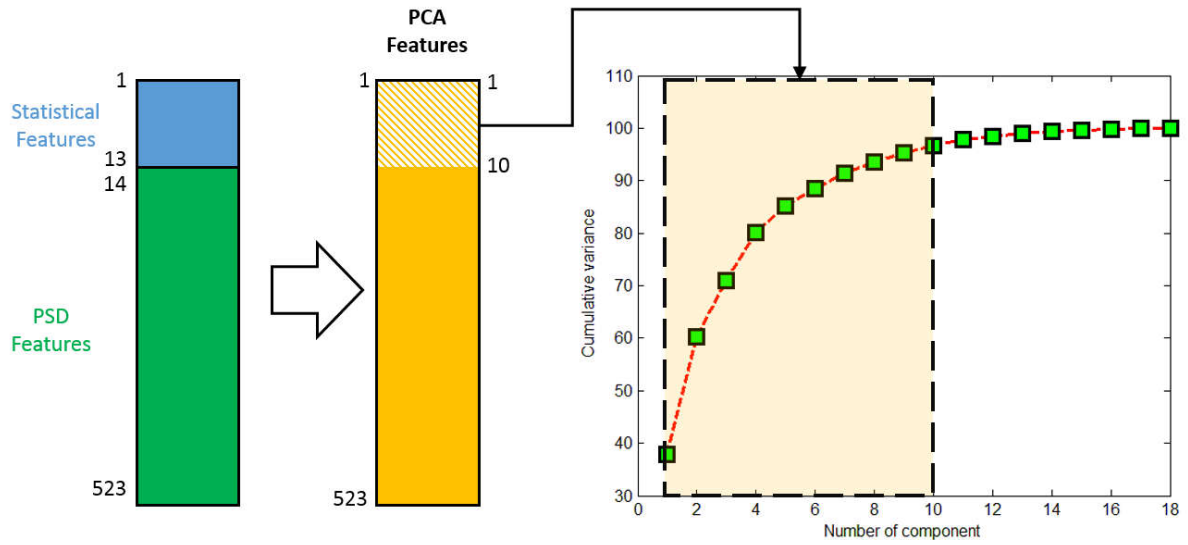


Figure 4.7 – Cumulative percentage explained by the principal components. Statistical and PSD features, in blue and green respectively, are converted into PCA components whose 10 first components can recreate 95% of the original features

In order to visualise how the vibration signal is perceived by the classifier, the first 25 PCA components for the vibration generated by each of the four types of floor can be seen in Figure 4.8. The four mosaic images represent 11 windows (columns) consisting of 25 components (rows) where a dark red colour indicates a value of 1 and a dark blue colour indicates a value of 0.

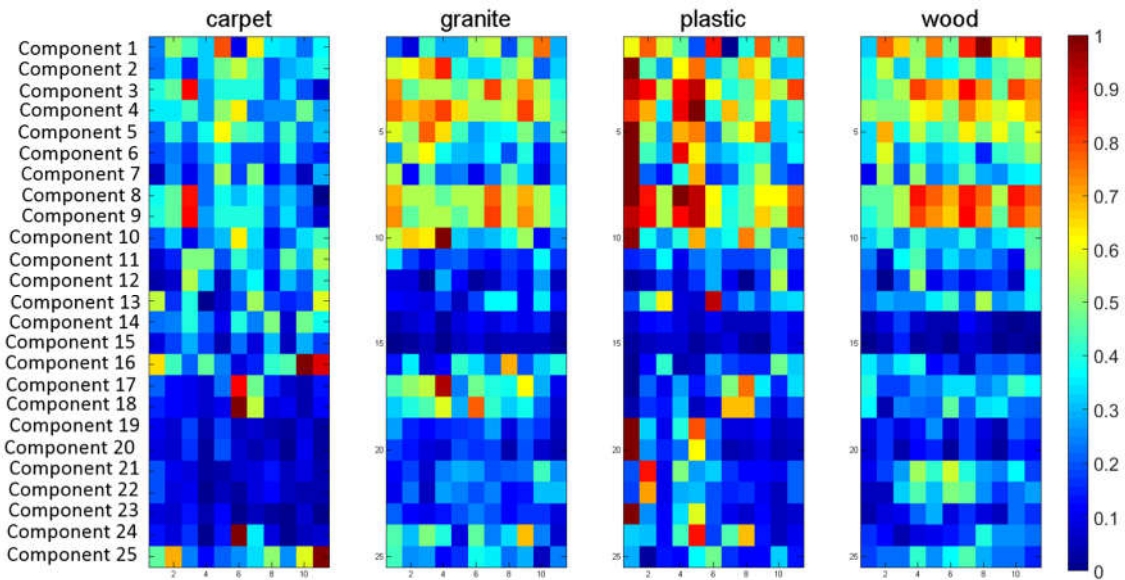


Figure 4.8 – PCA feature comparison between the four types of floors. The four rectangular patches represent the first 25 PCA components of 11 sample windows for each floor. Values are visualised as colours, where zero is dark blue and one is dark red

4.4 Training and Classification

Three different classification techniques were used to determine whether a linear or non-linear approach would be more suited for classifying this data. The first is Extreme Learning Machine [132] (ELM), a feed forward neural network work model that uses the Moore-Penrose inverse to efficiently calculate the weights of its output layer. For a set of data points composed of an input feature vector and output label vector (x_i, t_i) :

$$\begin{aligned} x_i &= [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n \\ t_i &= [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^m \end{aligned} \quad (4.1)$$

Where n and m represent the number of dimensions/features of input data points and the output label vector for that input point, respectively. The classification process uses an activation function $g(x)$, in this work $g(x) = \text{sig}(x)$, to create a new label vector based on the linear relationship between input x_i , w_i , b_i and β_i , expressed by Eq. (2):

$$\sum_{i=1}^N \beta_i g(w_i \cdot x_j + b_i) = t_j, j = 1, \dots, N \quad (4.2)$$

The number of hidden neurons is defined by L . w_i is the weight vector between the input neurons and the i^{th} hidden neuron, b_i is the bias of the i^{th} hidden neuron and β_i is the weight vector between the output neurons and the i^{th} hidden neuron. Eq. (2) is used for generating the label vector T during data classification and can be re-written in matrix format (Eq. (3) below) in order to simplify the calculations:

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \dots & g(w_L \cdot x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_1) & \dots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L} \quad (4.3)$$

and

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}_{N \times m}$$

which yields the final Moore-Penrose solution for the B matrix:

$$\beta = HT \quad (4.4)$$

ELM has proven to be very effective on non-linear classification [133, 134] and represented a good alternative for future real time surface classification applications. Different training sessions with different numbers of neurons were used to identify how it would impact the final classification result.

Other classifiers used, such as the Linear Discriminant Analysis (LDA), Quadratic Linear Discriminant (QDA) and Support Vector Machine (SVM), were selected to test if the data could be linearly separated in PCA space. These results would also allow benchmarks to be set for comparison with other methods in the future.

The last classifier used was the Gaussian Mixture Model (GMM), which represented a Bayesian approach for separating the data. With one single Gaussian mixture per class, initial results using 60 components yielded up to 98% accuracy on individual speed sets.

4.5 Results

4.5.1 Classifier Performance

The last stage of the floor classifier's development was selecting which classifier would best suit the set of features presented. A cross-validation test was devised using all data collected for the four floors (including all speeds and manoeuvres on each floor) in a 9-to-1 training to test ratio.

Next, the settings for the classifiers were determined according to how much they would impact the time required during classification. With the LDA and QDA, no adjustment was necessary. The GMM used only one mean and variance per components whereas the SVM used 10 support vectors. For neural networks, like the ELM, the number of neurons sometimes affects the accuracy of the classification at the cost of processing time, therefore, ELM used four different amounts of hidden neurons: 150, 200, 250 and 300. Figure 4.9 shows the results of the cross-validation test.

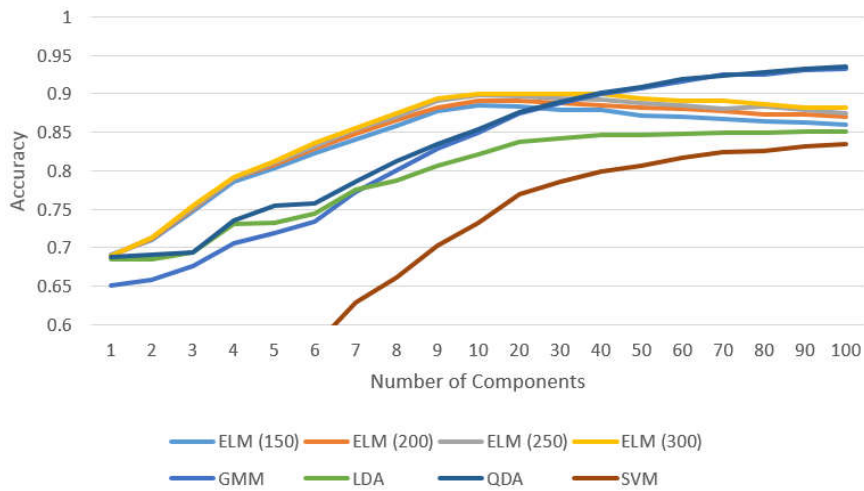


Figure 4.9 – Final overall classification using all data points from all recorded speeds and movements combined. ELM shows a higher classification accuracy with only 10 data components compare to the other classifiers

The compassion graph shows that for the GMM, LDA, QDA and SVM, the accuracy increases with the number of features (components) used. However, ELM reaches its plateau around 20 features and starts to overfit around 30 features. The overfitting of ELM classifiers is a known drawback of the technique that can be overcome with variations of the ELM itself or tuning of the features used for training. In this experiment, the ELM with 200 hidden neurons demonstrated 87.5% accuracy in classifying all four floors with a classification time of 650 microseconds, thus making it a good solution for real-time classification. From a technical point of view, slower classifiers could be used, as only one floor sample is classified every 250 ms due to the moving window; however, using more resources than necessary could end up restricting resources at later stages of the software development of the assistive robot.

The ELM with 200 hidden neurons was selected as the final classifier for the floor detection, and its performance on the cross-validation test was presented as a confusion matrix, shown in Figure 4.10 below, in order to better visualise which types of floor are more often misclassified.

	Carpet	PVC	Granite	Wood
Carpet	92.91%	3.29%	3.8%	0%
PVC	1.2%	89.09%	4.62%	5.09%
Granite	0.4%	6.27%	91.05%	2.28%
Wood	0%	11.86%	2.69%	85.45%

Figure 4.10 – Confusion matrix generated using the Extreme Learning Machine classifier using 200 neurons

4.5.2 Real Scenario Validation

In order to validate the classification accuracy on a real-life situation, the robot crossed a hallway between offices with two different surface types to classify. A map of the space was reconstructed by the robot and can be seen in Figure 4.11.



Figure 4.11 – Map of the robot's trajectory on the floor; green colour indicates carpet and blue indicates plastic surface

The carpet floor can be seen in green, and the plastic floor in blue. The robot was controlled using a wireless Xbox360® controller, where the maximum speed was set to 0.5 ms^{-1} . The trajectory started on the carpet on the right side of the map and traversed into PVC before returning back to the carpet area. At the moment the robot reached the division between floors, the operator recorded the transition using the controller by pressing a button. Doing so ensured an accurate recording of the ground truth floor transition in the timeline of the robot.

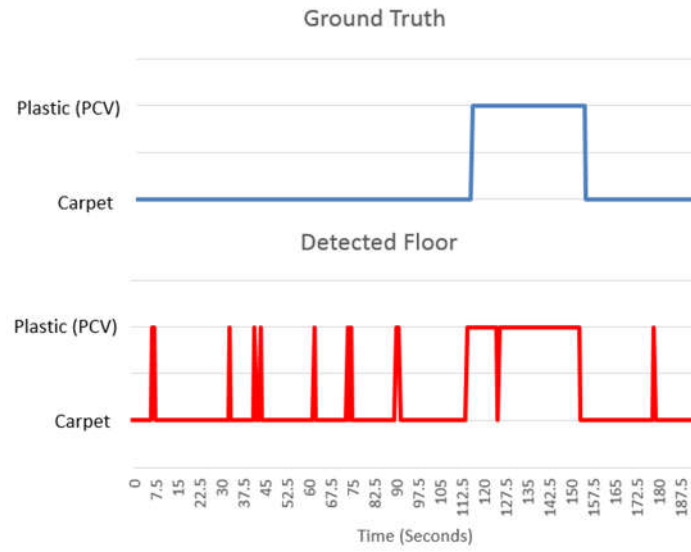


Figure 4.12 – Recorded floor classification over time. The detected floor (red) is compared to the ground truth (blue), where the peaks and dips on the red plot represent mis-classifications

4.6 Conclusions

In this chapter, a new method has been presented for accurately classifying different surfaces employing a single ELM neural network using a one-dimensional vibration signal. Four types of flooring commonly found in a hospital environment were used for the training, where different manoeuvres at different speeds were performed to collect vibration data from each of them. The classifier managed to compensate for extra noise added by the omni wheel rollers, with no pre-processing required. The classifier was able to correctly identify the floors with an overall accuracy between 85% and 92%. The overall accuracy of the classifier is ultimately correlated to how close the current speed of the robot is while manoeuvring on the floor compared to speed of the robot during data collection. At very low speeds, and at the robot's top speed, the classification accuracy fell under the 50% mark, which indicates that more data could be used for further training of the classifier.

Chapter 5

5 Dynamic Mapping

5.1 Introduction

Environment mapping is one of the most quintessential characteristic of mobile robotics, and perhaps the hardest task for a robot to accomplish on its own. There have been several works presented in Chapter 2 which demonstrated how a robot can effectively and accurately create a map of the environment as it is crossed; however, it is important to mention that most algorithms designed for simultaneous mapping and localisation do not take into account moving obstacles.

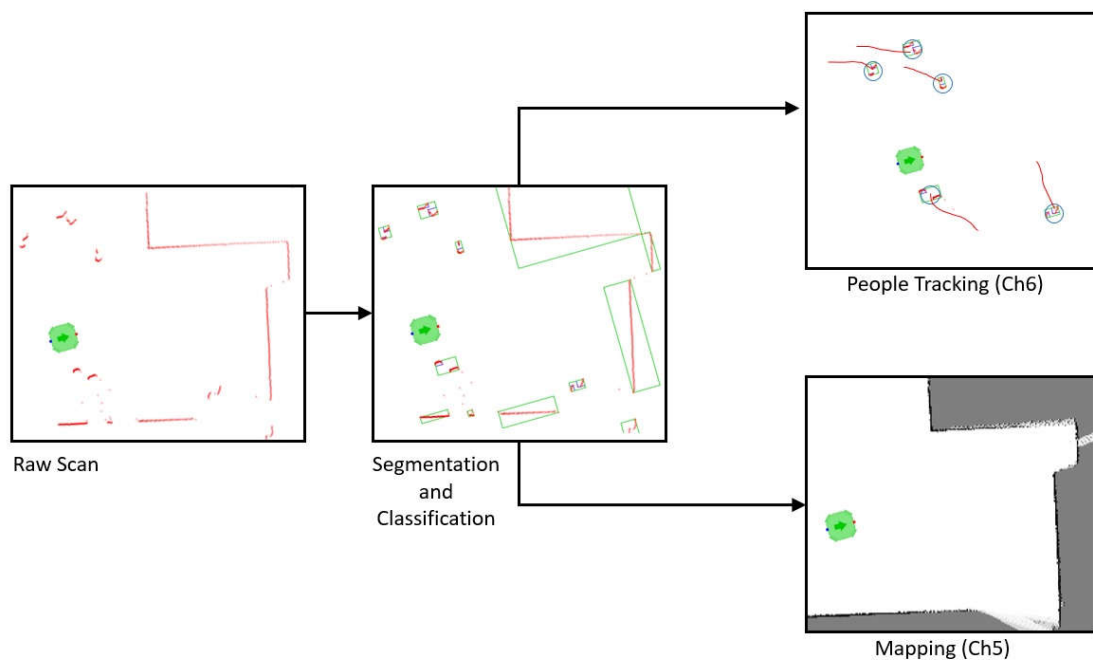


Figure 5.1 – The raw scan is segmented in to into clusters of nearby points that are then classified as a person, and used for tracking, or as part of the background, and used for mapping

Amongst the mapping algorithms developed, a few that take dynamic obstacles into account, but the shape of the object itself is ignored. This results in detecting people walking simply based on their constant movement. If a person remains still prior to the robot entering the room, this person would

most likely be registered by the robot as part of the environment. A possible improvement on the current dynamic mapping would be to detect what shapes, detected by the sensors, correspond to a person, and disregard them when creating the map.

This chapter proposes a method for adding a pre-processing stage to the Lidar scan data pipeline, designed to separate people from the background by detecting people's legs on the scan. A diagram of said pipeline is depicted in Figure 5.1. The raw scan is processed into groups of segments that are classified according to their shape as either "person" or "background". The segments classified as the "person" are further processed to provide a tracking of the moving persons (to be covered in Chapter 6). The remaining segments of the scan represent the background and are sent to the mapping module to be used by the SLAM algorithm.

5.2 HectorSLAM

HectorSLAM was selected as the primary mapping algorithm for this research due to its higher accuracy compared to other recent methods [31]. This approach is based on optimising the laser scan end-point alignment with the current map created up to that point in time; this idea is based on the image processing template matching using the Gauss-Newton method presented by [135]. Each new scan is projected onto the map based on the robot's last known position and orientation and the scan is iteratively aligned with the rest of the map. Once the alignment is completed, the difference in rotation and translation between the robot's current and previous position is checked against a threshold to determine whether the robot has moved enough to update the map.

A major drawback with using a finer resolution for the occupancy map for scan alignment, like 0.025 m per cell for example, is that it tends to lose alignment with the map, and by extension the robot's relative position if the robot moves too quickly. HectorSLAM solves this problem by introducing different levels of detail for the occupancy map and aligning the scan from the coarsest to the finest level. The end position and orientation achieved from a coarser level is carried out as the starting alignment position for a finer level. Figure 5.3 shows an example of the three levels of resolution for the same part of a map.

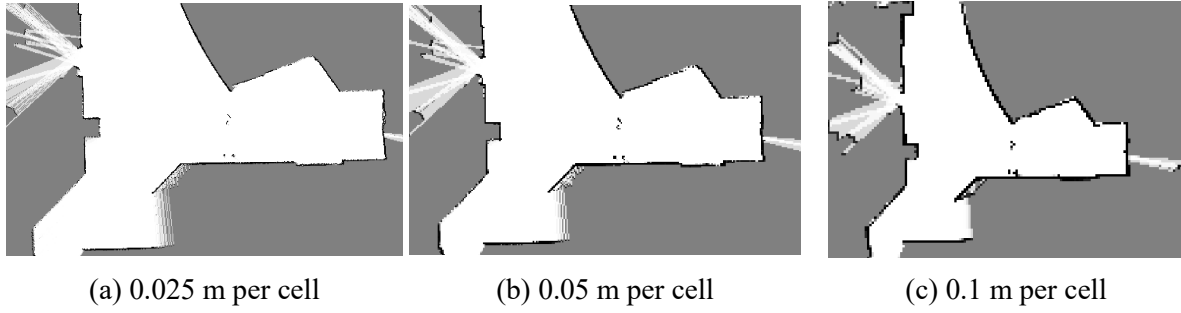


Figure 5.2 – Different resolution maps created using HectorSLAM. The larger the cell size, the lower the level of detail of the map

The first scan is imprinted on all three levels directly, and all subsequent scans use the current cell values to adjust the scan's position relative to the map.

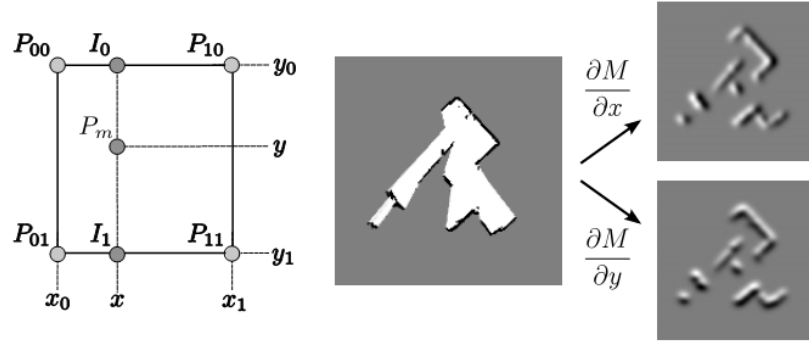


Figure 5.3 – Bilinear interpolation of the occupancy map [31]

The process starts from the coarsest to the finest level of detail in order to prevent loss of convergence. The Gauss-Newton process relies on using the map value and the derivative of the interpolated cell where the measured scan ends. The actual value of the cell shown in Figure 5.3 (a) is calculated by (5.1):

$$\begin{aligned}
 M(P_m) \approx & \frac{y - y_0}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} M(P_{11}) + \frac{x_1 - x}{x_1 - x_0} M(P_{01}) \right) \\
 & + \frac{y_1 - y}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} M(P_{10}) + \frac{x_1 - x}{x_1 - x_0} M(P_{00}) \right)
 \end{aligned} \tag{5.1}$$

Where $M(P_m)$ is the value for an interpolated point, P_m , which is calculated via a bilinear interpolation of its closest points on the occupancy map, P_{00} , P_{01} , P_{10} and P_{11} . The derivatives for Eq. (5.1) can be calculated by equations (5.2) and (5.3) below:

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y - y_0}{y_1 - y_0} (M(P_{11}) - M(P_{01})) + \frac{y_1 - y}{y_1 - y_0} (M(P_{10}) - M(P_{00})) \quad (5.2)$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x - x_0}{x_1 - x_0} (M(P_{11}) - M(P_{10})) + \frac{x_1 - x}{x_1 - x_0} (M(P_{01}) - M(P_{00})) \quad (5.3)$$

The goal is to calculate a rigid transformation $\xi = (p_x, p_y, \theta)$ that minimises the error in Eq. (5.4).

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^n [1 - M(S_i(\xi))]^2 \quad (5.4)$$

This transformation effectively aligns the laser measurements with the relative position of the robot:

$$S_i(\xi) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \quad (5.5)$$

Function $M(S_i(\xi))$ returns the probability value at the cell indicated by Lidar scan $S_i(\xi)$ at sensor position ξ . The algorithm seeks to minimise the alignment error by finding a $\Delta\xi$, given an initial ξ .

$$\sum_{i=1}^n [1 - M(S_i(\xi + \Delta\xi))]^2 \rightarrow 0 \quad (5.6)$$

Using the Taylor Series expansion in the implicit variable $M(S_i(\xi + \Delta\xi))$ in Equation (5.6), we derive

$$\sum_{i=1}^n \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^2 \rightarrow 0 \quad (5.7)$$

Which can be minimised by setting the partial derivative with respect to $\Delta\xi$ to zero, thus producing:

$$2 \sum_{i=1}^n \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right] = 0 \quad (5.8)$$

Solving $\Delta\xi$ in Equation (5.8) yields the Gauss-Newton equation for finding the absolute minimum error:

$$\Delta\xi = H^{-1} \sum_{i=1}^n \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T [1 - M(S_i(\xi))] \quad (5.9)$$

Where:

$$H = \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right] \quad (5.10)$$

The map gradient can then finally be calculated by the following derivative:

$$\frac{\partial S_i(\xi)}{\partial \xi} = \begin{pmatrix} 1 & 0 & -\sin \theta_{s_{i,x}} - \cos \theta_{s_{i,y}} \\ 0 & 1 & \cos \theta_{s_{i,x}} - \sin \theta_{s_{i,y}} \end{pmatrix} \quad (5.11)$$

5.3 Leg Detection

In the process of selecting which leg detection method to use for this research, the method proposed by [44] was initially considered due to its advantages previously discussed in detail in Chapter 2. This method is effective for detecting independent segments, rather than associating two leg segments as one person. Detecting people from the leg segments requires one extra step to associate the pair segments to one person on the scan. When one of the legs cannot be seen, it adds an extra level of difficulty to the association step; therefore, a simpler solution was adopted in this research.

A newer method based on generic distance-invariant features, proposed by [136], uses a built-in clustering step that connects nearby segments before the classification step. This not only simplifies the overall detection algorithm but also allows it to detect other shapes such as walkers and wheelchairs as “people”.

The feature histogram from every cluster of points in the scan is then classified as either a “person” or “background” by a binary Adaboost classifier [137]. The initial training of the classifier involved manual labelling of the clusters by a human expert, but due to the large number of labels required, the final training process used recordings where the robot was stationary and a pre-determined polygonal area was created to determine where people would walk. The clusters located inside the areas were classified as “person” and those outside as “background”.

5.3.1 Segmentation and Clustering

Each newly acquired scan is an array of 872 points (in our Lidar configuration) where each value of the array is the distance in meters from the Lidar to its respective collision with the laser beam. By knowing the angle between the beams and where to start counting around the circumference of the

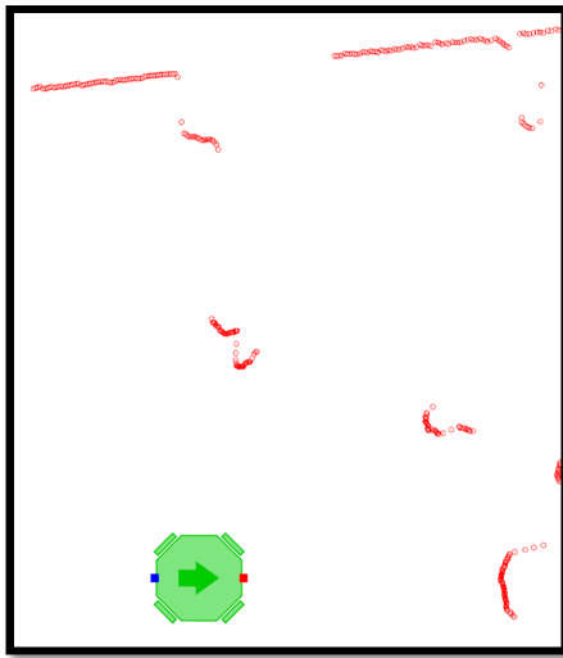
Lidar, the world coordinate positions can be calculated by simple trigonometry, as shown in Figure 5.4 (a).

Once the world coordinates are calculated, the points are then split into segments. Each segment starts with a point n and adds point $n+1$ if their Euclidian distance is shorter than a threshold (10 cm in our implementation). If the distance is larger than the threshold, the current segment is closed and the next points are added to a new segment. At the end of the segmentation process, only a stream of points is left belonging to the same segment (Figure 5.4 (b)).

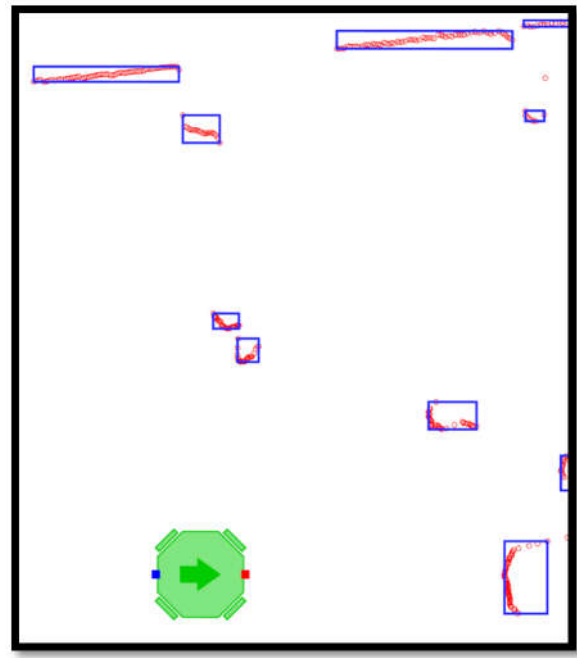
The next stage is clustering close-by segments in case they are legs of person standing. This stage reduces the overall complexity of the tracking system by reducing the number of tracked segments to one. It is simpler to combine two leg segments into a cluster and track that cluster than track each individual leg segment and try to associate both trajectories as one single person.

The original segmentation method proposed in [136] uses Delaunay Triangulation to identify close segments and combine them into one single cluster. A uniform-grid was used instead of the Delaunay Triangulation for its higher performance while inserting and accessing nodes in its data structure. Whilst a normal Hash table is boundless, as input coordinates can be truncated, therefore fitting any coordinate in a finite number of cells, a uniform-grid has delimited boundaries and uses the input coordinates of the objects as their actual locations in the table.

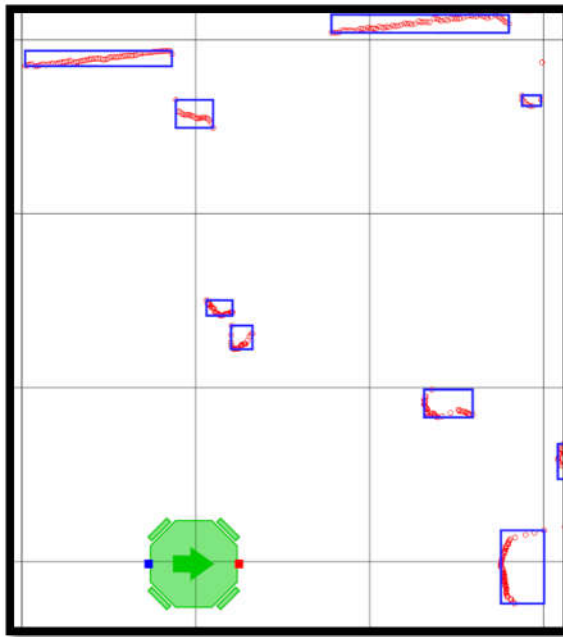
In this implementation, a two-dimensional uniform-grid with virtual size of 200 x 200 meters and cell resolution of 0.5 x 0.5 m was used. Each cluster of points is treated as a particle whose input coordinates to the grid is equal to the mean coordinates of its points — the centre of mass. Once the centre of mass is calculated for each segment, it is rounded to the closest grid point and added to the uniform-grid. The $O(1)$ complexity of accessing the uniform-grid outperforms the Delaunay triangulation in search time.



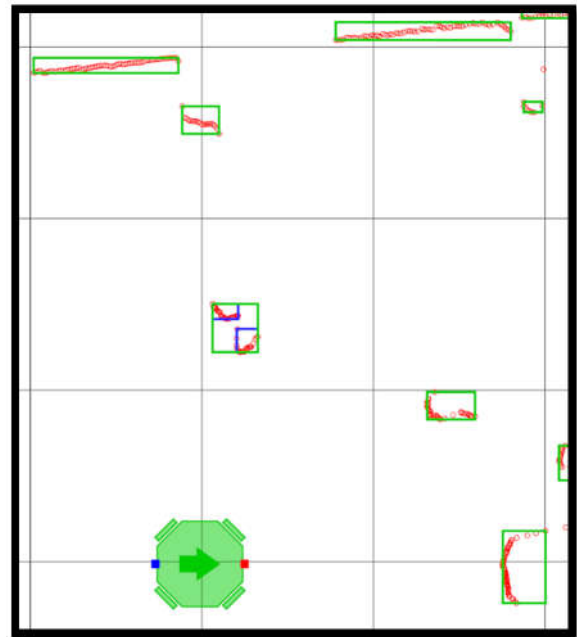
(a) Original scan



(b) Segmentation by distance



(c) Grid occupancy



(d) Clustering

Figure 5.4 – Step-by-step of segment clustering algorithm using a uniform-grid. Segments are separated and added to the uniform-grid before being clustered into the final clusters representing possible detected people.

In order to verify this claim, both methods were timed in the code along one entire map leg detection scan. The timing includes creating a brand-new data structure for each of the clusters detected in the scan. The results in Figure 5.5 show that the uniform-grid is more than twice as fast as the Delaunay triangulation under the same conditions for clustering segments.

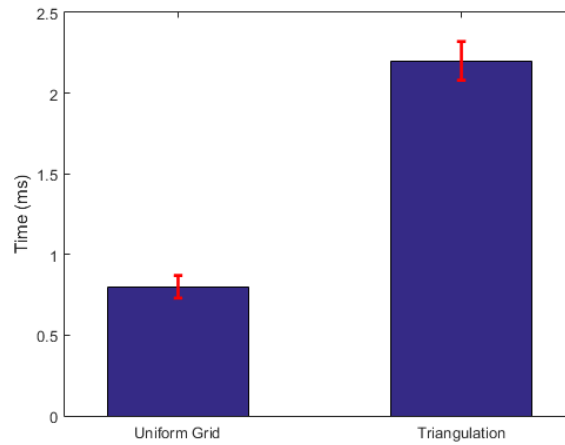
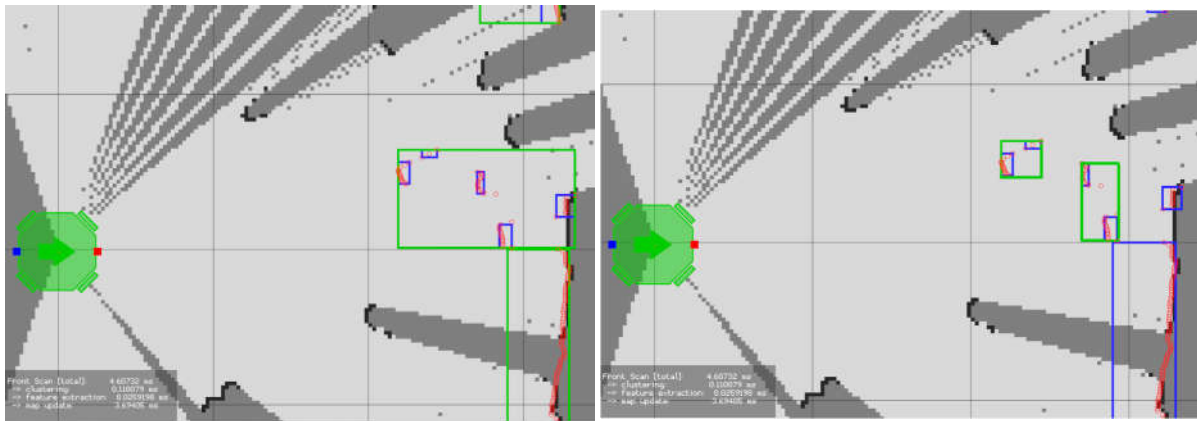


Figure 5.5 – Speed comparison between uniform-grid clustering and Delaunay triangulation clustering in terms of speed

The clustering process, however, creates the undesirable effect of combining legs of multiple people as one single target. This effect can be seen in Figure 5.6 (a) when two people are walking side-by-side at the right side of the image, and all their legs are clustered together.



(a) Two people wrongly added as one on the right

(b) Two people correctly separated

Figure 5.6 – Two people walking side-by-side are identified as one single individual if normal distance clustering is used

A simple solution found to this problem is limiting the size of each cluster to two segments. This is based on the assumption of the fact that the majority of people moving around will be standing on their two legs. The search-for-neighbours in the uniform-grid finds all the neighbours within the radius threshold and adds only the closest neighbour to the cluster. That cluster is then finalised and the algorithm moves to the next one. The result of the implemented solution can be seen in Figure 5.6 (b).

In practice, this solution does not work every time but it greatly minimises the number of false positives. In the case of single-leg segments or leg segments from multiple people still clustered together

passing through to the tracking stage, it is later dismissed by the multiple object tracker, which is covered in Chapter 6.

5.3.2 Cluster Classification

The last stage of the pre-processing pipeline is classifying each of the segments as a *person* or *background*. The classifier is trained using the same set of features proposed by [136]. These features consist of the values of small histograms containing the distance between each point in the cluster to an imaginary axis, perpendicular to the robot's line of sight. This method is illustrated in Figure 5.7 below.

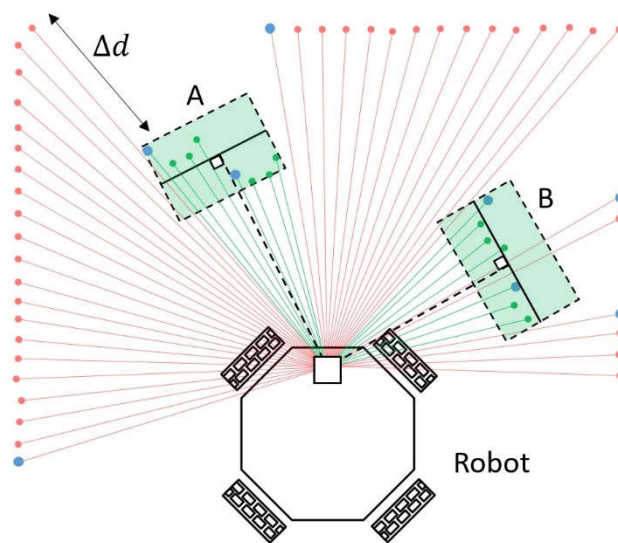


Figure 5.7 – Leg detection based on features from 2D clusters. Green areas A and B represent where the cluster of points represent legs are located. The blue dot represent the first point e each of the segment clusters.

The diagram above depicts a scenario where two people, A and B, are standing in front of the robot in a small room. The red lines and green lines represent Lidar laser beams that touch the background and the people in the scene, respectively. Each blue point represents the beginning of a cluster of points and the variable Δd represents the distance between points used to determine when the next point belongs to a new cluster – already explained in previous subsections. The mean point of each cluster of points is calculated and an axis is constructed (the solid black line in each rectangle), which is perpendicular to the vector between the Lidar and the mean point (the dotted black line). For simplicity, only two green rectangles are shown, one for each person, but each cluster of points passes through the same procedure.

The distance between each point within the cluster is projected onto the cluster's axis as the Y value, and a histogram of eight features is created along the X-axis, with width equal to the difference between the projections of the first and last point onto the axis. Examples of histograms belonging to legs in different positions can be seen in Table 5.1.

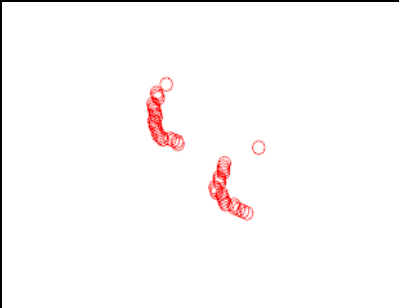
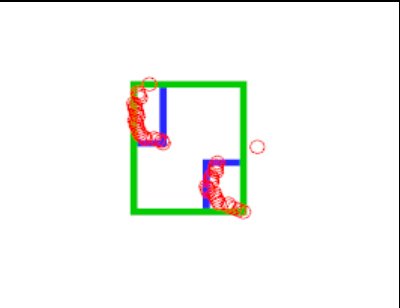
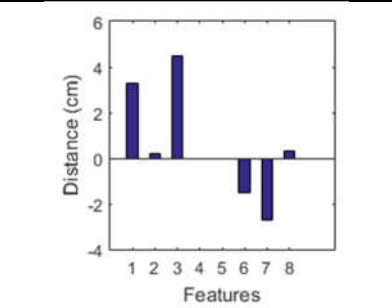
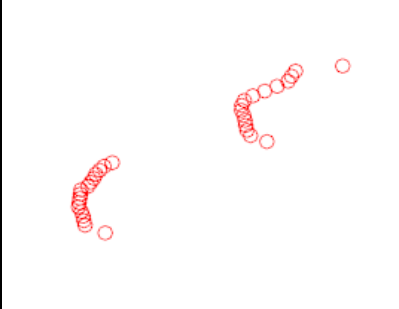
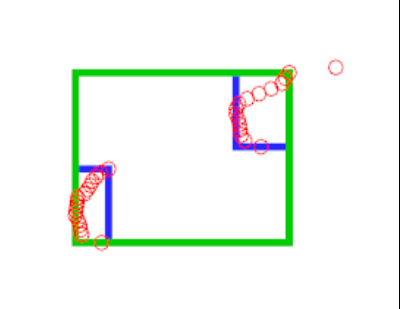
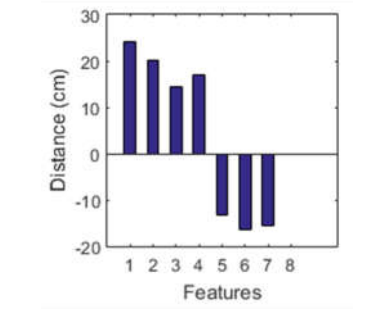

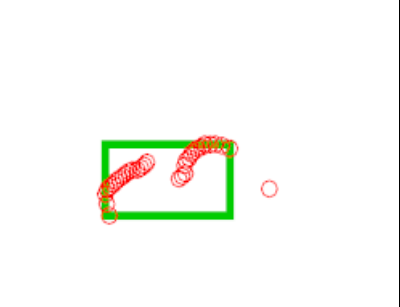
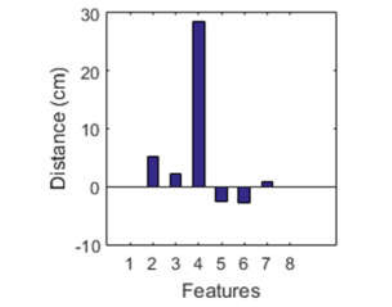
Leg segments on Scan	Segments Clustered	Feature Histogram
		
		
		

Table 5.1 – Examples of type of features for different representations of leg arrangements

The first column shows the red circles representing the shape of the legs. The second column shows the first and second stages of clustering where the blue rectangle represents the consecutive points in a sub-cluster, and the green rectangle represents the cluster composed of nearby sub-clusters. The third and final column shows the feature histogram extracted by the described method.

5.4 Results

5.4.1 Detecting people

A total of 173 frames, comprising of 566 clusters were manually labelled as either “person” or “background” by the author and used to train and test the Adaboost classifier described in [136]. The result of the ROC curve can be seen in Figure 5.8.

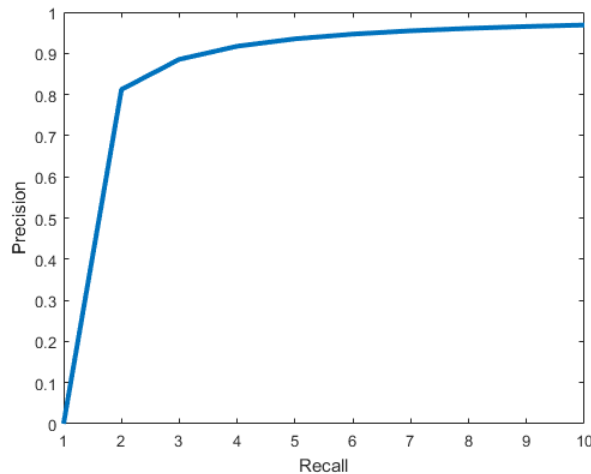


Figure 5.8 – Recall to precision graph (ROC) for leg detection using the Adaboost classifier

The training set was split into ten subsets where nine subsets were used for training, and one for testing. The following confusion matrix in Table 5.2 shows the results of the validation.

	Person	Background
Person	90.91%	9.09%
Background	6.8%	93.2%

Table 5.2 – Confusion matrix for leg classification

It is important to mention that these tests were performed with only two classes, as the original work presented in [136] already demonstrated that this method works similarly well for classifying not only legs, but also wheelchairs, walkers and crutches.

5.4.2 Improvement on Mapping by filtering persons in a real-world scenario

As mentioned at the start of this chapter, dynamic obstacles in a highly-populated environment can obstruct the Lidar, causing the robot to miscalculate both its relative position and its orientation during

the mapping stage. In order to evaluate the possible improvements in mapping accuracy by introducing the leg-detection pre-processing stage, scan data was collected in a real dynamic environment. The venue in which the data collection took place was an exhibition centre comprised of a suite of rooms on the second floor of the London-based Royal Academy of Engineering, in which an excess of 100 people were in attendance. A picture of the event can be seen in Figure 5.9.



Figure 5.9 – An actual room in which, whilst occupied, data was collected for verifying the possible improvements of the proposed new mapping method

The robot was manually guided across several connected rooms within this facility, during which delegates were socialising and roaming from room to room. Most of these spaces were outfitted with an array of tables and chairs in which wide open spaces were left in the centre of the rooms for people to circulate. The robot was driven around the second floor for a 45-minute duration in order to avoid disrupting the event attendees.

The challenge of this particular scenario was that, with numerous people surrounding the robot at any given time, its Lidar field-of-view was completely obstructed at times. What this meant in practice was that no wall segments could be detected. The inability to detect fixed features in the background represents a perennial challenge for mapping algorithms, rather like driving a car in the fog, as the circulating people are interpreted as part of the environment — even when the robot is stationary. One solution to this problem is to use odometry to prevent drifts on the robot's relative-position estimation. However, adding odometry messages would have the effect of increasing the complexity of the messaging system, when the aim is to simplify it.

The data was then processed using two versions of HectorSLAM: the original unadulterated version and that which is proposed, namely an altered version of HectorSLAM which introduces leg-detection at the pre-processing stage. Due to security restrictions and intellectual property concerns, it was not possible to obtain either an accurate layout or a scaled floorplan of the Royal Society of Engineering's Prince Philip Room. Nonetheless, it can be seen that the robot's navigation loop is closed, as demonstrated by the spatially-coherent version; Figure 5.10 (b); of the proposed Hector SLAM with leg detection method. The results of both versions can be seen in Figure 5.10 which follows:

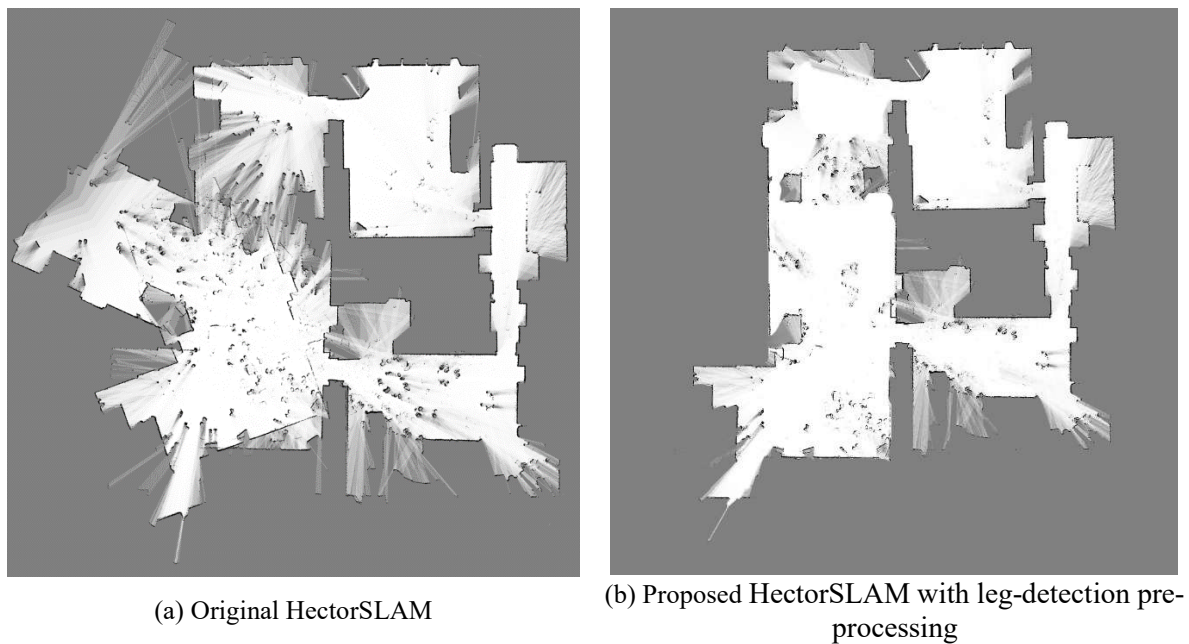


Figure 5.10 – Side-by-side comparison between two versions of HectorSLAM: (a) Original HectorSLAM and (b) Proposed HectorSLAM with leg-detection pre-processing

The first noticeable difference between the two maps is the spatial coherence of the generated map in (b). The original HectorSLAM used the raw scan data to recreate the map, which resulted in a loss of orientation and position in the main rectangular room at the left of the map. This is due to the large number of people surrounding the robot at certain points in its path, which were also added as parts of the map various locations as small artefacts in the middle of the white spaces. On the other hand, the proposed leg-detection pre-processing stage allowed HectorSLAM to recreate a more realistic representation of the 2nd floor of the building. Not all people were correctly identified, as demonstrated by a few artefacts in the middle of the white spaces, but all the removed scan segments representing the

legs were removed, thus allowing HectorSLAM to focus almost entirely on segments that represented walls and tables to create the map.

5.5 Conclusion

This chapter presented a new method for combining SLAM and machine learning for mapping crowded environments. The method improves upon HectorSLAM by adding the ability to identify people, moving or stationary, in the environment and excluding their signature from the mapping process.

First, the Lidar data was segmented, then clustered into groups of points that are likely to represent legs of people walking around. The clustering stage was able to correctly cluster leg segments together when people are walking at least 0.5 m apart but tended to group multiple people together when they walk at closer distances. A solution to reduce the number of incorrectly grouped segments was to create clusters with only two segments (as only people walking/standing with two legs were considered in this case), starting from the closest to the furthest segment in a connected row during the clustering phase.

Next, a feature histogram was created based on the distance between the cluster's points and their projection on an imaginary axis perpendicular to the robot's line of sight. An Adaboost classifier was used to classify the clusters as either a *person* or *background*, and the segments representing the background were used to map the environment. A real scenario with tens of people in a social event was used as a possible worst-case scenario to demonstrate how the modified HectorSLAM, with a leg-detection pre-processing stage, compared against the original HectorSLAM. Unlike the original method, the proposed method was able to successfully close the map loop and the final map represented fewer artefacts created by people standing by and roaming around.

Chapter 6

6 Dynamic Navigation

6.1 Introduction

Future roles assigned to the new mobile assistive robot may include following nurses to and from patient rooms, guiding patients and visitors across the ward, and delivering meals to rooms. These scenarios will require efficient real-time path planning to both avoid collision and react to any changes in the environment in a timely manner. This is largely because people will be constantly moving around the robot and changing their paths. Such matters represent a considerable challenge for the assistive robot due to the unpredictability of the environment and the limited amount of processing power available to search for optimal paths through a dynamic crowd.

Chapters 4 and 5 investigated new methods for classifying the floor and mapping a populated environment. The data collected during the experiments performed throughout the research of those chapters was acquired by the prototype robot under the remote control of a human operator, the author. Once a solution for mapping a dynamic environment was achieved, the next stage was to enable the robot to autonomously navigate the dynamic environment without colliding with anyone or anything. To that end, this chapter presents two improvements on current dynamic navigation methods aimed at reducing the overhead processing requirements of tracking its position in the map over time and updating its path according to changes in the environment. In addition, an improvement in performance to the localisation system is proposed by using the scan-alignment technique from HectorSLAM previously introduced in Chapter 5 in a state machine combination with Adaptive Monte Carlo Localisation (AMCL).

6.2 Position Initialisation State

During the assistive robot's normal operation cycle, it is expected to be switched on from its charging station, perform its tasks and finish its routine by returning to the same charging station. As the position and orientation of the charging station is fixed in the real world, the robot always knows where it is after being switched on, which allows it to move to its path planning phase and start following its pre-calculated path.

In the event of an error that requires the robot to reboot or an emergency shutdown triggered by pressing the emergency button, the robot would restart at an unknown position. As a safety measure, an initialisation stage was created to check whether the robot was powered on at its charging station or at an unknown location. If the location is unknown, the robot then performs a quick scouting routine to get its approximate location and orientation on the map.

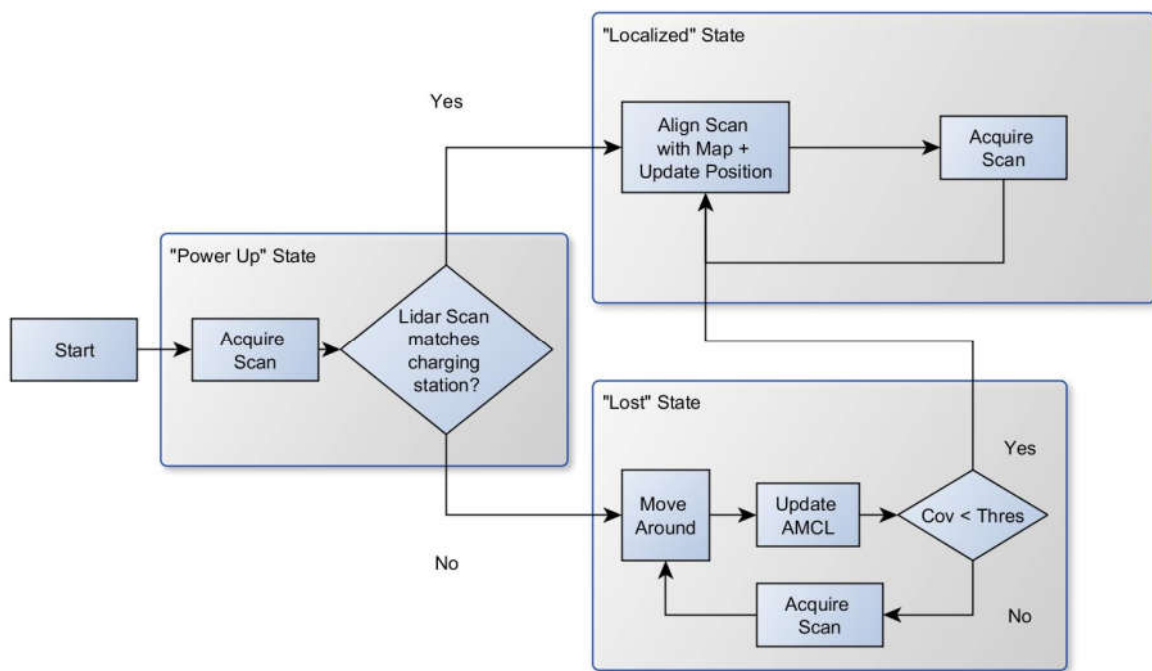


Figure 6.1 – Assistive robot's main localisation routine. The continuous scan matching is used when the robot knows where it is, otherwise, the AMCL method is used to located the lost robot

Figure 6.1 shows a flowchart of the robot's operation. Upon being initialised, the robot acquires a laser scan of the environment and compares it a model scan taken at the charging station. The scan is pre-process and app clusters classified as legs are removed. If the scans are similar enough, the robot

assumes it started at the charging station and the system moves to the “Localised” state; otherwise it assumes it could be anywhere on the map and moves to the “Lost” state.

At the “Localised” state, the robot uses the HectorSLAM’s scan alignment sub-routine to update its current position and orientation on the map. After the update, a new scan is acquired and the process repeats itself. Throughout the testing and development, the robot did not lose track of its position while moving or being stationary, not even once.

At the “Lost” state, the robot uses the Adaptive Monte Carlo method (AMCL) [138] to estimate its position and orientation by roaming around, scanning the environment using the Lidar. The position estimation is updated by reshuffling the particles around the environment and increasing the weight on the particles whose simulated scan more closely match the current scan been detected. The algorithm itself can be described by Eq. (6.1), and is divided into two steps:

$$p(x_t|z_{1:t}, u_{0:t-1}) = \alpha p(z_t|x_t) \int p(x_t|u_{t-1}, x_{t-1}) p(x_{t-1}) dx_{t-1} \quad (6.1)$$

- Prediction Step: Given a previous action, u_{t-1} , a new sample is drawn for each previous sample according to the robot’s movement model, represented by the distribution $p(x_t|u_{t-1}, x_{t-1})$.
- Correction Step: The new observation z_t is added to the sample set by resampling the weighted likelihood $p(z_t|x_t)$ of observing z_t given the robot’s current position x_t .

Details of the full set of equations and how the algorithm is executed are provided on the original paper. When the overall particle distribution reaches a minimum variance (determined empirically), it is assumed that the robot’s current position is close enough to the real-world position, and so the localisation state switches from ‘Lost’ to ‘Localised’.

6.3 Continuous Localisation Using Scan Alignment

When the robot is operating inside the ‘Localised’ state, only the scan-matching sub-routine from the HectorSLAM is used to update its position and orientation. At every new scan, its position and orientation are updated with no odometry information required. The scan alignment method has been

found to be reliable for speeds under 1.8 m/s and rotation speeds around 120° per second, which is above the assistive prototype robot's normal operation speeds.

The AMCL method on the other hand, despite being a very common solution for continuous localisation, only provides a position and orientation update when the robot moves beyond a certain distance or rotates beyond a certain angle threshold. Real-time updates are provided by odometry estimation, unlike the HectorSLAM scan matching, which provides updates at every new scan. AMCL also requires a large number of particles in order to increase its chances to converge its estimate to the robot's real position, which increases the processing cost for each update.

6.4 Tracking Position and Movement of People

In Chapter 2, different techniques for tracking multiple moving objects were presented and discussed, and the Probability Hypothesis Density (PHD) filter demonstrated a good trade-off between ability to track multiple objects and the complexity of its calculations compared to the standard Kalman Filter approach. Although there have been improvements over the years on the filter itself, the initial proposed implementation of the Gaussian Mixture PHD filter (GM-PHD) [139] was found to be more than adequate to handle tracking of multiple people moving around the robot.

The PHD filter is designed based on the Random Finite Sets (RFS). Due to the noisy nature of the Lidar sensor and the limitation of the leg segment clustering algorithm, some of the measurements may be false positives. The finite set of n elements X_t represents all the states x_t for each tracked obstacle at time t . Likewise, the finite set of m elements Z_t represents all the sensor measurements z_t at time t . Supersets $\mathcal{F}(X)$ and $\mathcal{F}(Z)$ represent all the possible states and measurements respectively. The finite sets can be expressed as the following:

$$X_t = \{x_t^1, \dots, x_t^n\} \in \mathcal{F}(X) \quad (6.2)$$

$$Z_t = \{z_t^1, \dots, z_t^m\} \in \mathcal{F}(Z) \quad (6.3)$$

The state of the system at time t is estimated from its previous state $t - 1$ by the expression:

$$X_t = S_{t|t-1}(X_{t-1}) \cup B_{t|t-1}(X_{t-1}) \cup \Gamma_t \quad (6.4)$$

Where

- $S_{t|t-1}(X_{t-1})$ represents the spatial movement of the targets that survive from time $t - 1$
- $B_{t|t-1}(X_{t-1})$ represents the birthing of new targets between times t and $t - 1$
- Γ_t represents the birthing of new targets at time t

The evolution of the states of the system is denoted by the “multi-target transition density” [139] $f_{t|t-1}(X|X_{t-1})$, and the “multi-target likelihood” [140] $h(Z_t|X_t)$. Both terms are used in Eq. (6.5 and its posterior estimation in Eq. (6.6. Notice that X and Z denote sets of particles of variable size from iteration to iteration.

$$p_{t|t-1}(X_t|Z_{0:t-1}) = \int f_{t|t-1}(X_t|X)p_{t-1}(X|Z_{1:t-1})\mu_s dX \quad (6.5)$$

$$p_t(X_t|Z_{0:t}) = \frac{g_t(Z_t|X_t)p_{t|t-1}(X_t|Z_{1:t-1})}{\int f_{t|t-1}(X_t|X)p_{t-1}(X|Z_{1:t-1})\mu_s dX} \quad (6.6)$$

The PHD filter recursion reduces the inherent complexity of combining multiple distributions, commonly used in the multiple target Bayesian tracker, by propagating the density of the distribution to further time steps.

$$v_{t|t-1}(x) = \int g_{s,t}(\zeta)f_{t|t-1}(x|\zeta)v_{t-1}(\zeta)d\zeta + \int \beta_{t|t-1}(x|\zeta)v_{t-1}(\zeta)d\zeta + \gamma_t(x) \quad (6.7)$$

$$v_t(x) = [1 - p_{D,t}]v_{t|t-1}(x) + \sum_{z \in Z_t} \frac{p_{D,t}(x)h_t(z|x)v_{t|t-1}(x)}{v_t(z) + x \int p_{D,t}(\zeta)h_t(z|\zeta)v_{t|t-1}(\zeta)d\zeta} \quad (6.8)$$

6.5 Path Planning Methods

Path planning is the area concerned with determining future positions of the robot in order to avoid physical obstacles. Robot arms use path planning to predict the positions of the actuators and space whereas mobile robots, like the assistive robot proposed in this research, use path planning to plot all waypoints to be followed in order to reach the destination without bumping into anything or anyone.

The path planning methods to be discussed below do not track obstacles in real time but rely on third-party systems to provide information as to where all the objects are.

The A* search heuristic is a commonly used path-planning method for finding the lowest cost path in a graph search, due to its robustness and simple implementation. It is, however designed for static environments, as recalculating the entire path every time someone moves renders the method prohibitively expensive for real-time applications.

A variation of the A* algorithm that reduces the processing time by sacrificing the shortest path guarantee is the weighted A-star. This variation simply uses a coefficient to change the final cost for each cell, thus biasing the search towards a more directed path towards the goal. Figure 6.6 shows a comparison between the results obtained from different weighing coefficients.

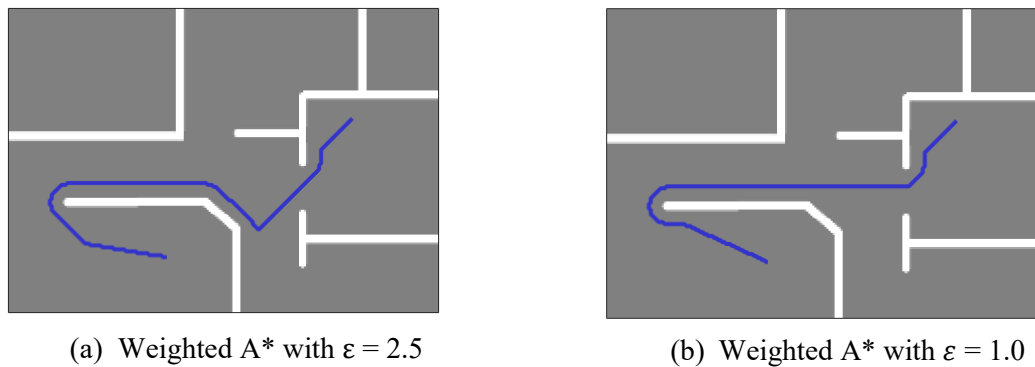


Figure 6.2 – Comparison between different weights for A*. The higher the coefficient epsilon, the faster the path is to compute, but the less optimal it becomes

A coefficient, ϵ , of 1.0 (original A* configuration) produces a path of 12.66 m in 171 ms, whereas a coefficient of 2.5 produces a path of 13.68 m in 100 ms. A higher coefficient produces a longer path in less time, and a coefficient of 1 will produce the optimal path at the longest time. The exchange between path length and time required to process it is a desirable trade-off when pursuing a real-time path planner.

This idea was further explored in the Anytime Repairing A* (ARA*) [141] method, which uses a series of Weighted A* searches to produce a path solution within an allocated time. The first search has the highest coefficient and requires the least amount of processing time to produce a sub-optimal

solution with lowest processing requirements, followed by subsequent searches with shorter and shorter path lengths and respectively higher and higher processing times.

6.5.1 Safe Interval Path Planning (SIPP)

The Safe Interval Path Planning (SIPP), introduced in [53], presented a different method for handling dynamic obstacle avoidance during the path planning phase. The SIPP method, based on the HCA* algorithm [142], executes its path search in current and future states of all cells on the map. SIPP, like the HCA*, guarantees an optimal solution at the expense of a longer processing time. It assumes that the robot is able to wait in place without falling over and that inertial constraints, such as acceleration and inertia, are negligible. In the case of the assistive robot, this assumption is valid as the robot is capable of standing in place on its four wheels and is able to reach its target cruising speed in hospitals in a few hundred milliseconds, which are negligible in journeys which typically last tens of seconds.

SIPP vastly improves upon HCA* by reducing the search space from a large array of continuous fixed-time intervals of an unchanged state to a single time interval representing the availability for a cell between two time periods. A safe interval is defined as the continuous period of time when the cell is not occupied by any moving obstacles. The time is divided into time steps which are concatenated into a single time interval defined by the first and last time steps in which the cell is unoccupied. A collision interval is defined by the first moment an object moves into a cell to the last moment when he leaves. If the obstacle remains in the cell until the end of the search time, the interval's duration is assumed to be infinite.

Figure 6.3 shows a simplified example of how SIPP creates time intervals based on the availability of the cells at any given time. In this example, the robot and the moving obstacles are treated as particles that occupy one cell at time. The robot, located at (6,5) has to reach (3,5) without colliding with any of the moving obstacles O_1 and O_2 , which are moving east and south, respectively. The world state for each of the eight time steps, T_0 to T_7 , shows how the robot moves out of the way of O_1 at T_3 and moves out of the way of O_2 at T_5 , finally reaching its destination at T_7 .

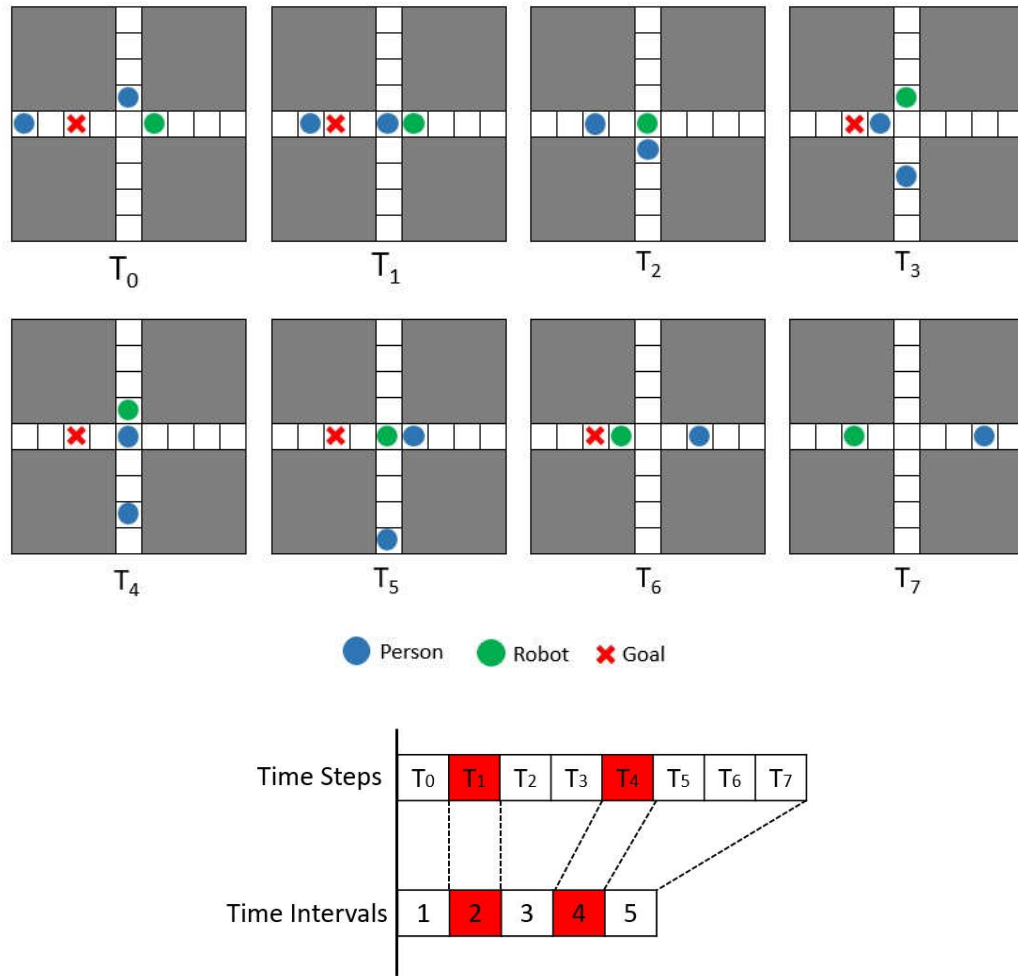


Figure 6.3 – Collision diagram showing robot and moving obstacles in their respective trajectories, and key points indicating delimitations of collision interval over time

The entire manoeuvre happens in eight time steps in this example, but in the real world, each time step would be 250 ms in duration, which considerably increases the total number of states to search for a path. The SIPP method compresses the total number of states for each cell-to-time interval, shown at the bottom of Figure 6.3, where the central cell's availability over time, shown in eight time steps, is compressed to five time intervals. Cells that are never occupied by any moving obstacles have only one time interval because all time steps are labelled as “free”.

In a real-world scenario, making the real dimensions of a cell large enough to accommodate the physical body of the robot and the people would generate a low-resolution representation of the environment. However, in order for the robot to be able to plan its path amongst moving obstacles, a finer resolution of the space state of the map is required. The solution is to represent the body of the

robot with multiple cells of smaller size, like in the original SIPP method. Figure 6.4 (a) below shows an example of all the cells that a robot, denoted by a circle with radius equals to 0.3 m, would occupy in a 10 cm per cell resolution. At every time step, a new filled circle is drawn on the next layer of the volumetric space representing the total state-space. If the robot is moving on one direction, the overlapping of the occupied cells over time by the robot could take the shape, for example, like the one shown in Figure 6.4 (b) below.

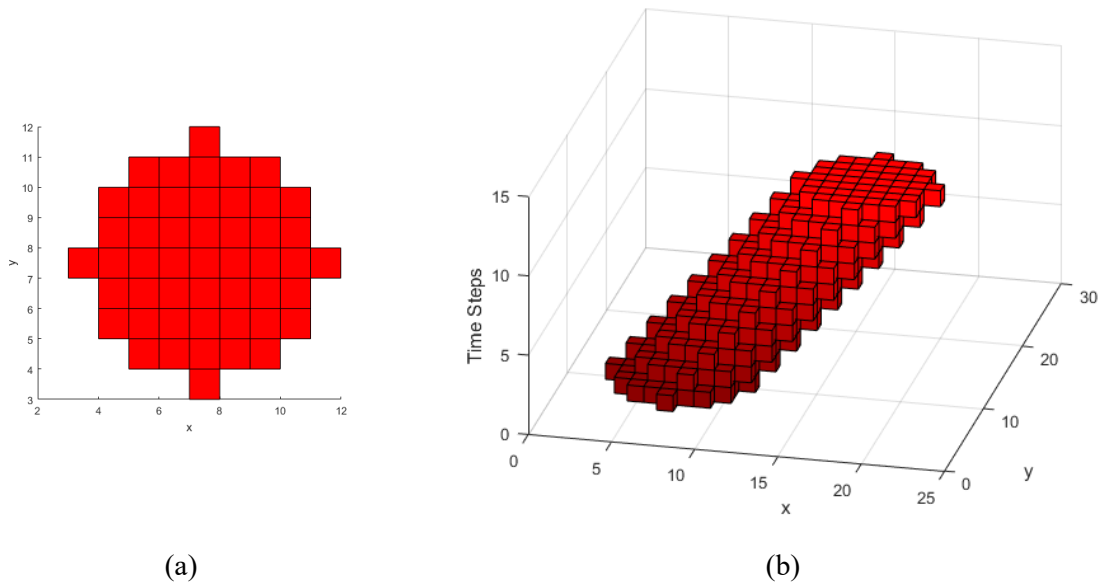


Figure 6.4 – Real-world representation of the robot's footprint discretised into cells (a) and same footprint, depicted over time, showing the robot moving in a straight line (b)

The original SIPP method used a robot equipped with a differential drive for its results, which requires using a lattice planner to generate all viable actions at any given point. These viable actions are defined by the robots' own kinematics, and in the case of differential drive, it is a non-holonomic configuration. The prototype assistive robot, however, is equipped with a holonomic base that allows it to move in X and Y whilst being able to rotate at the same time. Given that the prototype robot does not move at excessive speeds, dynamics of the movement are not considered and therefore the robot can perform any of the orthogonal movements on a dime. This means that the lattice planner is not necessary to determine the viability of its motions, but rather its viable actions can be approximated by the cells around its position at any given time — provided it is not moving at its top speed.

SIPP uses a modified version of A* that incorporates searching in future states for each of the cells without changing the A* core heuristics. The modified search algorithm can be seen in Figure 6.5.

```

1 g(sstart) = 0; OPEN = []
2 insert sstart into OPEN with f(sstart) = h(sstart);
3 while(sgoal is not expanded)
4   remove s with the smallest f-value from OPEN;
5   successors = getSuccessors(s);
6   for each s' in successors
7     if s' was not visited before then
8       f(s') = g(s') = 1;
9       if g(s) > g(s) + c(s, s0)
10        g(s') = g(s) + c(s, s');
11        updateTime(s');
12        f(s') = g(s') + h(s');
13        insert s0 into OPEN with f(s');

```

Figure 6.5 – Modified A* Algorithm with Safe Intervals in place [53]

In the algorithm above, every state s in the map has a variable $g(s)$ that represents the path with the lowest cost from the beginning of the origin to the state s . The A* heuristic function $h(s)$ is an estimated cost from current state s to the goal of the robot. The cost of taking an action, like moving to an adjacent cell or waiting in place, is defined by $c(s, s')$, where s is the state prior to the action, and s' denotes the final state.

Upon initialisation, the SIPP algorithm is run once to determine a valid path to the goal. The subroutine *getSuccessors*, shown in Figure 6.6, returns all possible actions that can be performed at each point in time, including the amount of time required to complete them.

```

1 getSuccessors(s)
2 successors = []
3 for each m in M(s)
4   cfg = configuration of m applied to s
5   m_time = time to execute m
6   start_t = time(s) + m_time
7   end_t = endTime(interval(s)) + m_time
8   for each safe interval i in cfg
9     if startTime(i) > end_t or endTime(i) < start_t
10      continue
11     t = earliest arrival time at cfg during interval i with no collisions
12     if t does not exist
13      continue
14     s0 = state of configuration cfg with interval i and time t
15 insert s0 into successors
16 return successors;

```

Figure 6.6 – “getSuccessors()” subroutine used in the modified A*[53]

In Figure 6.6, function $M(s)$ returns all the viable actions that the robot can take at any state s . Each action has a cost to it, which takes into account how long the robot has to wait before starting the acting as well as the how long it takes to execute it. The start and end times for a safe interval i are represented by variables $startTime(i)$ and $endTime(i)$. For every future state s' from a state s , a new successor state is generated, as long as the time in which the robot arrives and the time the robot leaves that state do not overlap with a collision interval. In order to create a path that arrives at its destination at the earliest possible time, each new action starts at the earliest possible time in its interval that is guaranteed not to enter a collision interval after the robot moves. If necessary, the robot has to wait in place until such time becomes available.

During path search using the original A* algorithm, a current path is replaced by another path whenever the combined cost of the most recent state indicates a shorter path. In the modified A* algorithm used in SIPP, the same principle holds, however, that the path with the earliest arrival time is the one considered the shortest path.

An improvement over the original SIPP method is the Anytime Safe Interval Planning (ASIPP) [54] which combines the near-real-time replanning capabilities of ARA* with the SIPP method. ASIPP relies on the same assumptions inherent to the use of SIPP, namely that the acceleration is negligible and that the robot can wait in place if necessary, and adds the use of a time horizon, originally introduced by the Time-Bound Lattice method [143], to limit the search depth of SIPP in order to comply with the time restrictions imposed by the ARA*.

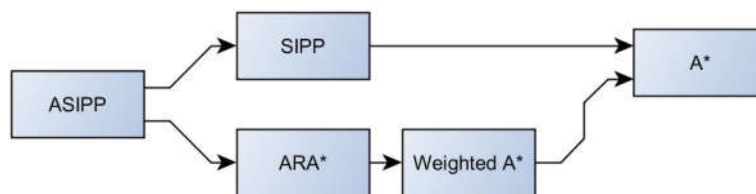


Figure 6.7 – Diagram showing the hierarchical dependency of the path finding methods discussed in this chapter.

The diagram shown in Figure 6.7 illustrates how SIPP relates to all the previously discussed path planning methods. ARA* requires Weighted A* in order complete searches in different time windows.

In turn, Weighted A* requires the original A* method with the addition of a coefficient to alter the cost for the cells. By combining SIPP and ARA*, ASIPP is capable of returning path solutions of various lengths in real time, depending on the available processing time.

One of the major drawbacks of the ASIPP is that, like its predecessor SIPP, it also requires knowledge of where each obstacle is and where they are going at all times. In the original published work, the final results were presented from a simulation where the robot had access to the position and velocity of all moving obstacles at all times. In a real-world scenario, like in a hospital, keeping track of every moving person at every accessible area would require a highly complex visual tracking system to be installed in each hospital.

6.6 Anticipated Improvement Over Previous Methods: Dynamic Safe

Interval Path Planning (DSIPP)

The proposed method for this chapter focuses on improving two major limitations found in both SIPP and ASIPP: The need to access a large memory volume to determine the availability of all map cells at all times, and the need for knowing the position and velocity of all obstacles on the map at all times. We introduce a collision detection based on detecting the intersection between the 2D footprint of the robot and the moving obstacles' current trajectory, represented by a vector in order to both remove the need for discretising the collision space and resuming the overhead cost of re-calculating the collisions whenever an obstacle changes its path.

6.6.1 Trajectory Update

At every update, the last 10 consecutive known positions of every visible moving person are used to determine the person's respective mean direction and velocity. This direction is considered the estimated trajectory vector, with origin set to the position where the person was first seen or the last time they changed their direction and/or speed.

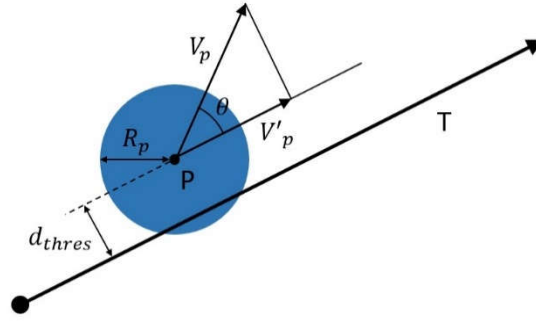


Figure 6.8 – Example of how a moving obstacle is tracked throughout its trajectory. The person is represented by the blue circle and their estimated trajectory is represented by the vector T

Figure 6.8 shows an example of how the moving obstacle is tracked along its estimated trajectory. The moving person is represented by the circle centered at P with radius R_p . The person's trajectory is represented by the trajectory vector T . As the person moves further along their path, the distance between the person and the estimated trajectory is the perpendicular vector d_{thres} , calculated by projection point O onto the trajectory T .

First, if the distance d_{thres} is found to be beyond 0.1 m, the trajectory is updated to reflect the current deviation from the estimated course. The threshold of 0.1 m was determined after many trials using varying distances for updates, ranging from 0.05 m to 0.25, and 0.1 m was found to be a good compromise between resilience to noise introduced by position estimation and accuracy in term of trajectory prediction.

Next, if the person changes their speed, V_p , the projected speed along the predicted path, V'_p , also changes, which can trigger another path-planning update if the new speed is 0.15ms^{-1} faster or slower than the original recorded speed of the person at the time the trajectory was predicted.

Finally, if new people are detected by the robot, or any of the previously visible people are lost for more than a few seconds, a path update is also triggered. The tracking algorithm used in this case, the PHD filter, plays a major role in determining if a new update should take place, as it keeps tracks of all visible moving obstacles and associates their positions over time, even when some readings are erroneous or missing. Since only the Lidar is used to track people legs, it is more probable that misclassifications happen when compared to using an image identification system.

One specific scenario that could represent problems for the path planning is when the robot is required to backtrack in order to let a person pass by, in narrow corridors, for example, and the line of sight between the robot and the person is broken. If a path planning update was set to be triggered whenever the current number of people being tracked changes, a “loop scenario” could be created accidentally in situations when the robot had to move away from the person’s view in order to let them through. This “loop scenario” was verified in the simulation and is illustrated in a simplified diagram in Figure 6.9.

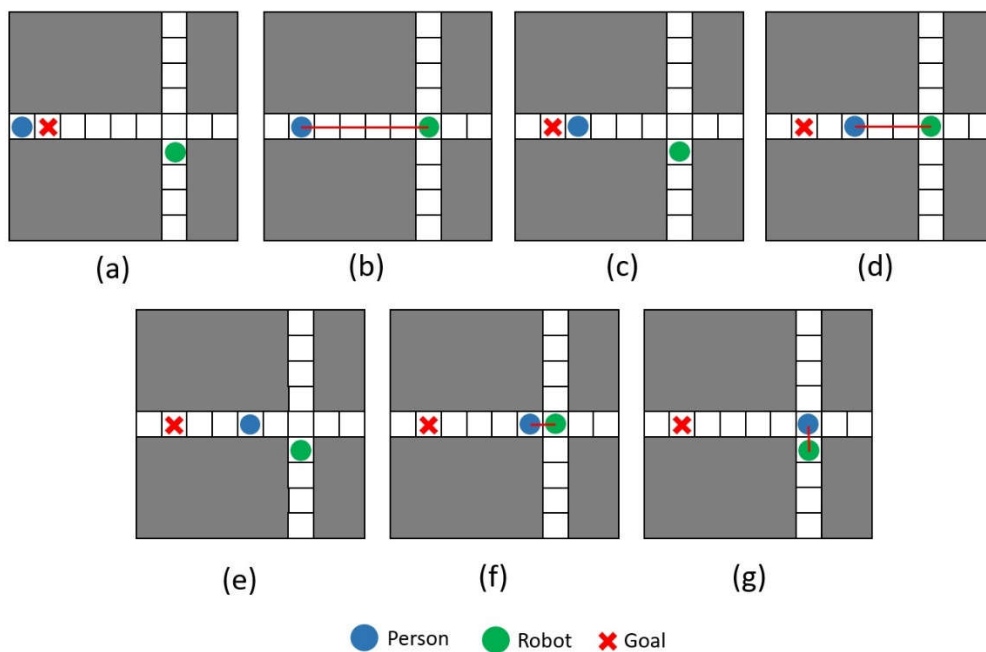


Figure 6.9 – ‘Loop scenario’ where the person moves in and out of sight, forcing the robot to re-plan and repeat. This is fixed by adding a wait condition before re-planning.

At (a) the robot has no visual contact with the person coming from the west and is about to turn the corner towards its goal at (2,5). When line-of-sight is created between the robot and the person at (b), shown as the red line, the new path plan is to move away from the person’s linear trajectory and wait for him to pass. However, when the robot moves to (7,4) at (c), a new path planning update is triggered and, since no people are in sight, the robot resumes its original path. At (d), the same scenario in (b) is repeated, and this causes the robot to move back and forth between positions (7,4) and (7,5). This repeats until the person is always in sight and the updates stop. In practice, the robot moves back and forth a few centimetres, just enough to occlude the Lidar sensor from the moving object.

This problem could be solved by simulating the person's movement when out of sight for long periods of time, assuming constant velocity, and re-matching their predicted and actual position. However, a simpler solution found was to disable any path update until the robot finishes its next waiting period. If implemented in Figure 6.9, after the robot detected a moving person at (b), its path plan would be to move out of the trajectory and wait, but all line-of-sight updates would be disabled until it finishes that wait time. In practice, this simple solution proved very effective in many different scenarios.

6.6.2 Geometric Collision Detection

The geometric collision detection is a proposed solution to reduce the time required to detect collision states on each cell by detecting the time and duration of each collision, thus providing collision-intervals directly. The original SIPP or ASIPP would generate a three-dimensional volume of cells (as the orientation of the robot is not a dimension, given the holonomic drive) with width and depth equal to the map's dimensions, and depth equal to the number of time-steps allocated to discretise the availability of each cell over time. Converting time-steps of a cell to time-intervals to be used by the A* algorithm would require traversing every single layer of the three-dimensional volume for each of the cells being checked on the map. In addition, the exact moment of collision is determined by the resolution of the time-step. Reducing the size of the time-step can increase the time accuracy of the collision; however, it also increases the amount of memory required and time taken to check all the extra layers of the map.

In comparison, the geometric collision detection is achieved by testing if the perpendicular distance between the robot and the moving obstacle's linear trajectory is shorter than the sum of both the robot's and the obstacles radii. This not only dismisses the need to transverse a three-dimensional volume of cell states, but also returns the exact moment in time the collision occurs. An example of how the geometric collision detection works is demonstrated in the figure below (6.10).

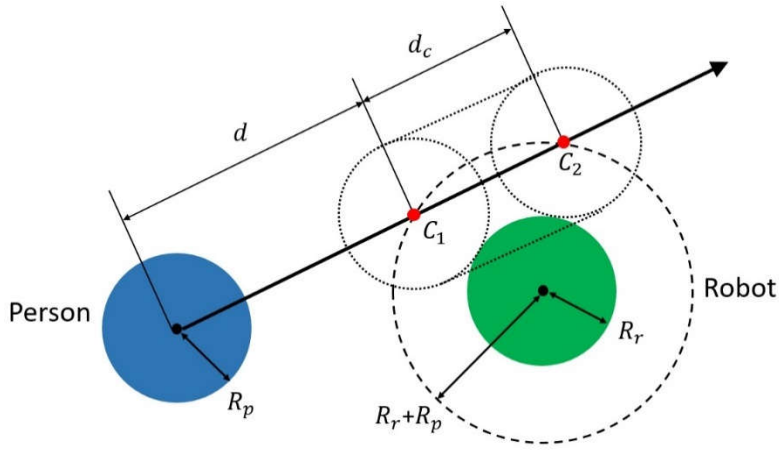


Figure 6.10 – Collision detection using a line and circle intersection method. The points C_1 and C_2 represent respectively the beginning and end of the collision between the robot and the person, should the person maintain their trajectory

In Figure 6.10, a person (blue circle) and the assistive robot (green circle) are seen near each other, where the vector starting at the centre of the blue circle indicates the direction the person is moving. The diameter of the circles represents the person and robot's collision distance. A collision is detected when both circles overlap. The position of the robot is not necessarily the actual position of the robot, but rather, a possible position of the robot in one of the map's cells at a point in time being tested by the A* algorithm. An initial collision is detected at point C_1 , and the collision continues should the person remain in their linear trajectory, until point C_2 . Points C_1 and C_2 can be calculated by detecting the intersection of an imaginary circle with radius equal to the sum of both the person and the robot's collision radii, $R_r + R_p$, and the estimated trajectory of the person. The trajectory is a vector with origin equal to the person's position when the trajectory was estimated. The respective points representing the person and the robot's centre are:

$$P_{person} = [P_x, P_y]$$

$$P_{robot} = [R_x, R_y]$$

Using these points, the following parameters can be calculated to detect where the collision(s) between the line and the circle happen, if any.

$$d_x = R_x - P_x$$

$$d_y = R_y - P_y$$

$$d_r = \sqrt{d_x^2 + d_y^2}$$

$$D = \begin{vmatrix} P_x & R_x \\ P_y & R_y \end{vmatrix} = P_x R_y - R_x P_y$$

The intersection solution is the same as a quadratic equation, yielding zero, one or two solutions, depending on the value of the delta function:

$$\Delta = (R_r + R_p)^2 d_r^2 - D^2$$

Where if $\Delta < 0$, there is no intersection and therefore no real solution; if $\Delta = 0$, the circle and the line are tangential; and if $\Delta > 0$, there are two intersection points. The x and y values for C_1 and C_2 can be calculated by:

$$x = \frac{D d_y \pm \text{sgn}(d_y) d_x \sqrt{\Delta}}{d_r^2} \quad (6.9)$$

$$y = \frac{-D d_x \pm \|d_y\| d_x \sqrt{\Delta}}{d_r^2} \quad (6.10)$$

Eq. (6.11 can return no points in case of no collision; one point – tangent circle to line – indicating an infinitesimally short collision between the robot and the person; or two points (the most common outcome), indicating a continuous collision over time. The duration of a detected collision can be defined as the time interval between t_1 and t_2 :

$$t_1 = \frac{d}{|V_{person}|} \quad (6.11)$$

$$t_2 = \frac{d + d_c}{|V_{person}|} \quad (6.12)$$

The duration of this continuous collision is equal to the collision-interval of this cell where the robot is placed. If more obstacles collide with the robot at that position, more collision intervals can be calculated and added to that cell's states. A comparison between the two methods can be seen in the example depicted in Figure 6.11, where two moving obstacles collide with the robot at different points in time.

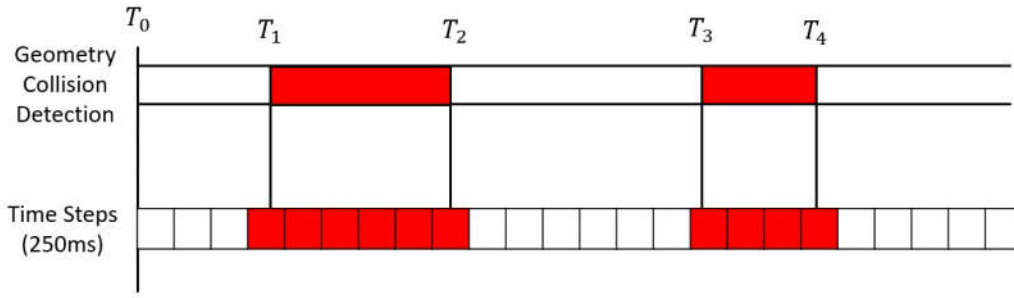


Figure 6.11 – Comparison between discrete time based and geometric based collision detection. The continuous time-step holds the entire duration of the occupied state, whereas the discrete method requires an iterating through each time step.

The diagram above shows how the geometric collision detection method (top) is able to accurately compress all the collision data in two time intervals during the collision detection stage. In contrast, the time-step discretised space (bottom) consists of 20 cells that need to be transverse and compressed into two collision intervals at the time-interval stage.

6.7 Experiments and Results

In order to quantify the difference in performance between the collision detection used in both SIPP and ASIPP, and the new geometric-based method used by the proposed DSIPP, a series of simulated test scenarios were created. The use of simulation was necessary to provide a deterministic repeatable scenario, whereas it would be virtually impossible to recreate the same states running the experiment in the real world.

The experiments were based on the scenario described in the state-of-the-art solution [54], which is constituted by a 300 by 300 cell area with various moving obstacles roam around. The time-step was set to 100 ms and the robot was instructed to cross an area avoiding all the moving obstacles. Path planning updates for both ASIPP and DSIPP were triggered by obstacles that moved in and out of sight. **Error! Reference source not found.** shows the scenario configurations used for the experiments.

In the case of many moving obstacles, obstacles further away would sometimes be obstructed by obstacles nearby in the same line-of-sight. Obstacles that moved in and out of line-of-sight were tracked using the PHD filter, but extra path planning updates would still be generated due to the high visually obstructed region around the robot. The robot and the obstacles can be seen as the green and red circles,

respectively. The blue path is the current path the robot has calculated to avoid all moving obstacles within sight, and the magenta trail lines represent the last position that each of the obstacles have been within 5 seconds. As the obstacles were set with different velocities, the length of their trails vary.

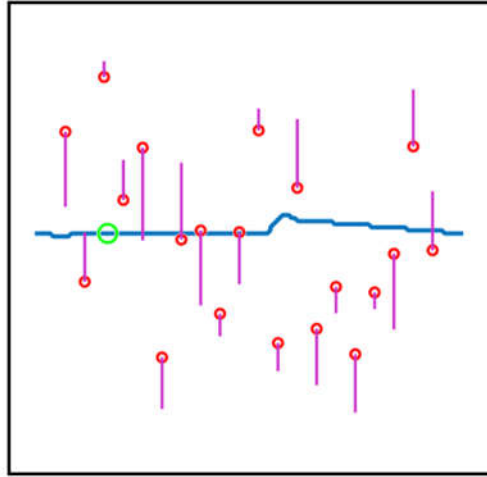


Figure 6.12 – One of the scenarios with the robot (green circle) avoiding 15 people moving across the map. The robot's calculated path can be seen in blue, and the person's trajectory vectors can be seen in magenta.

Two main variables were selected for their direct effect on the performance of both methods: the time horizon for ASIPP and the number of moving obstacles for the proposed DSIPP. The ASIPP introduced the concept of a time-horizon threshold as a limit to how far the robot can plan ahead before it has to re-plan its path. The larger the time-horizon, the longer it takes to populate and search the 3D volume containing the collision states of the cells before the time-interval stage. On the other hand, the number of moving obstacles directly affects the performance of DSIPP — instead of populating a volume with the state of every cell before running a weighted A* algorithm, each cell searched by A* has to be tested for collision with all visible obstacles. The higher the number of obstacles, the longer it takes to check every cell.

A combination of a number of obstacles, varying from 2 to 30, and time intervals, varying from 5 to 25 seconds, was used, and the time taken for planning each path was recorded for each run. The results for both ASIPP and DSIPP methods can be seen in Figure 6.13. The 3D plots show the number of obstacles and time horizon duration on the bottom X and Y axes, and the processing time taken to plan the path is shown on the Z-axis.

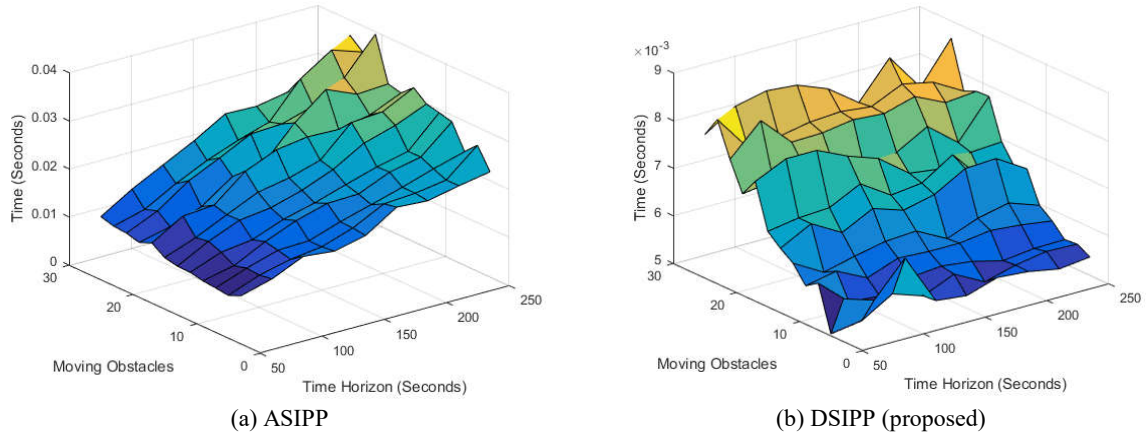
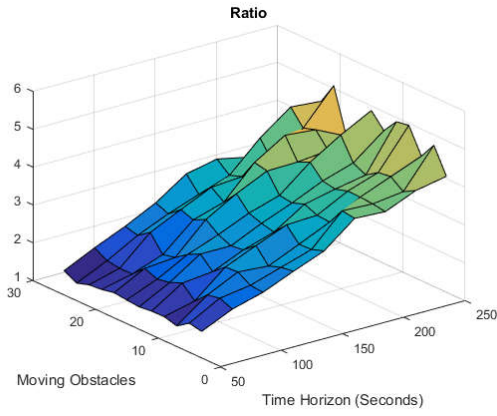


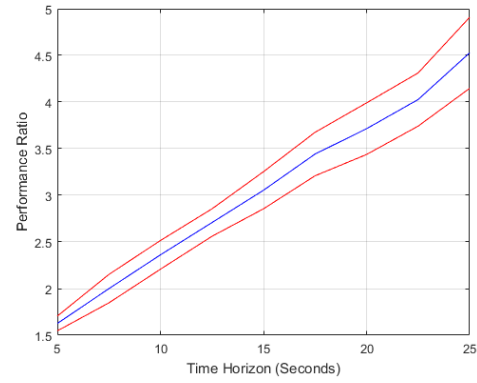
Figure 6.13 – Time taken for each update for each person that entered the robot's line-of-sight. ASIPP becomes slower as the time horizon increases, whereas DSIPP becomes slower as the number of visible moving obstacle increases

In Image (a), the ASIPP method, as the length of the time horizon increases, more cells need to be accessed for registering the movement of all moving obstacles. The number of cells that need to be accessed grows in a cubic progression to the number of time steps in the time horizon. With a time horizon of five seconds, all planning is completed within around 10 ms; however, in cases where the robot has to wait, 5 seconds is not enough. A 20-second time horizon was found to be more adequate to handle real and simulated scenarios with realistic corridor width. Additionally, the number of moving objects has little effect on the time required to re-plan, as the collision detection stage is done prior to the weighted A* path search.

In Image (b), varying the length of the time horizon has no significant effect on the proposed DSIPP method as it does not have a pre-process all collision cells before creating the time-intervals. Instead, DSIPP creates collision time-intervals for every cell, and checks them during the weighted A* by using the geometry-based collision detection introduced in 6.6.2. On the other hand, the time required to finalise the path search in the DSIPP method increases with the increasing number of moving obstacles. It is also possible to see that the longest processing time is still less than 10 ms, compared to the 40 ms of the ASIPP method under the same conditions. In order to better visualise the speed improvement between the previous and the proposed planning methods, two ratio plots were generated in Figure 6.14 below.



(a) Ratio plot 3D



(b) Ratio plot 2D – Side view

Figure 6.14 – Performance improvement between ASIPP and proposed DSIPP methods shown in a 3D plot (a), and on a 2D plot (b). The blue line and red lines on the 2D plot represent the mean and one standard deviation of the number of moving obstacles for each time horizon

The result of dividing the processing time from both methods is a 3D slope, seen at Image (a), that is mostly affected by the increase in length of the time horizon. This effect was expected as the DSIPP is not affected by the time-horizon, thus keeping its processing time nearly constant, while the time required by ASIPP increases. The number of obstacles that influenced the slope of Image (b) in Figure 6.13 does not appear to have any effect here due to the much smaller magnitude when compared to the time values in the ASIPP plot. Therefore, as the number of moving obstacles causes little to no variation, a 2D version of Image (a) from Figure 6.14 was created as Image (b).

In Figure 6.14 (b), the blue line indicates the mean value of the 3D plot on 2D dimensions (Ratio and Time Horizon) and the red line indicates one standard deviation of the combined 3rd dimension values. Figure 6.14 (b) shows a clear improvement in performance in favour of the proposed DSIPP method as the time horizon increases. At a 25-second time horizon, the proposed DSIPP method is 4.5 times faster than the ASIPP, thus making the DSIPP a more suitable candidate for real-time applications in dynamic path planning.

6.8 Conclusions

This chapter presents two contributions in the areas of real-time localisation and dynamic path planning. The first contribution is a way for combining the scan alignment sub-routine from HectorSLAM into a finite state machine that uses AMCL to locate the robot in case it is lost and

switches it over to the scan-alignment system when the robot is certain of its location. The scan alignment uses the data processed by the leg detection system (Chapter 5) which makes it less prone to drift when around many moving obstacles. This method also requires a fraction of the processing power compared to particle-based methods and doesn't require any odometry information, which saves space in the robot's internal messaging system.

The second contribution of this chapter is an improvement in both speed and memory usage on the dynamic path planning method, Anytime Safe Interval Path Planning (ASIPP). Tracking dynamic moving obstacles is achieved by using the Probability Density Filter (PHD), which is more suited for tracking many moving obstacles compared to combined Kalman filters of Joint probability distribution filters for its reduced processing overhead requirements. The key difference between the proposed Dynamic Safe Interval Path Planning (DSIPP) and SIPP/ASIPP is the removal of the pre-processing collision detection stage, where the cells that the moving objects occupy at any time must be populated into a 3D array with width and depth equal to the map dimensions, and height equal to the length of the time horizon in time-step units.

The new method dismisses any memory pre-allocation and instead generates direct time intervals from checking collisions with every visible obstacle at every cell evaluated by the Weighted A* algorithm. The collision detection is achieved by a simple circle-to-line intersection, which is both computationally inexpensive and provides a continuous value for the boundaries of the time interval. Several experimental simulations were run, varying the number of visible moving obstacles, and the length of the time horizon and the proposed DSIPP demonstrated to be between 1.5 times (at 5 second time horizon) to 4.5 times (at 25 second time horizon) faster than the previous state-of-the-art solution.

Future improvements to the DSIPP could be focused on the line-of-sight update condition. Currently, the "loop condition", in which the robot enters a re-planning loop when moving objects leave its sight in certain situations, is fixed by only allowing further path planning updates after reaching the next waiting period. A possible path for development could be to introduce an internal simulation of all

moving obstacles with the aid of the implemented PHD filter and synchronise its predicted state with its real state at every new scan.

Chapter 7

7 Sensor Fusion for Long Range Gaze Estimation

7.1 Introduction

Some of the roles the future hospital assistive robot could undertake include delivering goods to and from rooms and guiding patients and visitors in the hospital. Whilst delivering goods is a task that relies heavily on following a path and avoiding obstacles, guiding people would introduce a human element to the path-following and obstacle-avoidance problem. The assistive robot should, in theory, be able to verify if the person is not deviating from the path (adjusting accordingly if necessary), to wait for the person if they stop, and to assess if the person is paying attention to any instructions given on the screen. In addition, people following the robot would present various degrees of cooperation and cognition, which would make designing a robust method for guiding anyone in any scenario a challenge far larger the scope of this chapter. Instead, this chapter focuses on improving the perception of subtle visual cues that would help the assistive robot determine the person's intention during a task and adapt accordingly.

In Human Robot Interaction (HRI), gaze detection in the wild is a difficult challenge due to many environmental factors, such as lighting conditions, crowds, and hardware limitations of the sensors. In the reviewed literature, two main types of gaze detection were presented: Image processing and Infrared (IR) based. Image processing relies on detecting the position of the heads on the image, following the eyes' relative position on the face in order to estimate a gaze direction. These methods are computationally expensive and performance varies depending on lighting conditions. The gaze direction estimation is not very accurate and tends to be noisy due to the resolution of the image. Processing images from a high-resolution camera would require expensive parallel processing hardware to run it in real time. Methods that use IR are mainly hardware based and rely on illuminating the person's face with IR light to cause the pupils to reflect part of the light back, thus making the only the

eye visible on an IR camera. Detecting the position of the eyes in space is achieved through simple image-processing and trigonometric operations. These methods often only work on short distances (less than 60 cm) and require previous calibration, which render them difficult to implement on general populations.

An example scenario where a mobile assistive robot is guiding an elderly person through the hospital can be seen in Figure 7.1 above. In this scenario, the robot's sensory ability to detect the person's gaze would enable it to be more proactive in its actions and verify if any new on-screen instructions have been seen by the person.

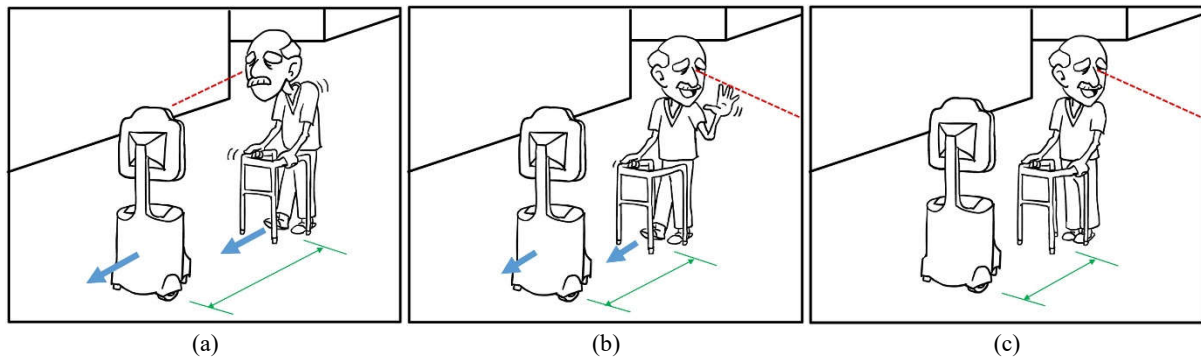


Figure 7.1 – Illustration of a scenario where the robot is guiding an elderly patient using gaze tracking to enhance its response: (a) Robot detects that the patient is looking at it and maintains course, speed and distance from the person; (b) If the person diverts their gaze for some time, the robot can detect the person's change in focus and reduces its speed to prepare to wait for the person in case they stop; (c) The robot waits for the person to look back at it before resuming guidance.

In Image (a), the robot is seen moving at the same speed of the person being guided, shown as the blue arrows, and maintaining a fixed distance, shown in the green arrow-determined spacing between them. At this state, the robot is also able to determine that the person is looking directly at it and any instructions displayed on its screen are visible to the person. If the person fixes his gaze somewhere else for more than a short period of time, as depicted in Image (b), it is safe to assume that the person is now focused on someone or some event besides the robot. Without gaze tracking, the robot would still be able to detect if the person slows down or stops (adjusting its own speed accordingly), but it would not know if the person is able to see the screen, read new instructions or even distinguish if any gestures are directed at the robot itself or someone else. In Image (c), both the robot and the person have come to a complete stop as the person seems to be engaged in a conversation with someone outside the

frame. The robot can wait for the person to finish his conversation and return focus to the robot, or perhaps even try to intervene if too much time has passed by moving to the centre of the person's line-of-sight.

This chapter proposes a hardware implementation of a gaze detector to be used in ranges beyond 60 cm by combining an IR camera, IR illumination and a depth sensor. The idea is to combine the eye detection by the IR eye-tracker with the real position of the head detected by the depth sensor. With two sources for the position of the eyes, the final position is expected to be less noisy and therefore more reliable. Several experiments were conducted to evaluate the accuracy and limitation of the hardware in a range of scenarios. The methods and results presented in this chapter were devised, planned and implemented collaboratively with fellow student; now Doctor; Stephen McKeague; the accompanying hardware solution was wholly designed by the author as a specific requirement of this research.

7.2 Gaze Detection System Design

The design of the gaze tracker was based on the successful implementation of the previous works presented in Chapter 2, and further expanded to incorporate a depth sensor to improve accuracy. Figure 7.2 shows the 3D CAD model of the eye-tracker built for the experiments. The complete gaze tracker shown on the CAD model consists of one camera with a visible light filter, three sets of IR lights (two off-centre and one on the camera), and one Kinect depth sensor. Both eye tracker and Kinect cameras are registered so that each pixel on the eye tracker camera can be projected on the Kinect's point cloud. A local linear interpolation is used in cases where pixels on the eye tracker camera do not directly correspond to pixels on the Kinects depth camera to ensure a smooth depth estimation.

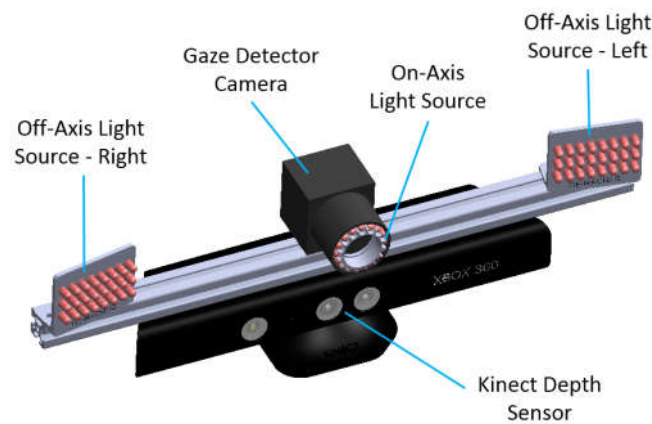


Figure 7.2 – 3D model of the final assembly version of the eye tracker. The IR camera was fixed on an aluminium extrusion, with the on-axis illumination source mounted inside the camera's lens' rim. The off-axis illumination sources are mounted on both ends of the bar, and the Kinect sensors is fixed at the bottom

The eye tracker has two groups of LEDs that alternate flashing between frames of the camera. The off-centre group consists of two panels of LEDs located at either end of the aluminium bar that generate an IR image of the subject when lit. The on-centre group of LEDs generate a nearly identical image as the off-centre LEDs', with the main difference being that the pupils appear much brighter. The Kinect depth sensor is used to acquire a 3D point cloud of the environment to be used during the sensory fusion stage. The actual construction of the eye tracker used 3D printed parts to firmly secure the cameras on the frame, as any translation or rotation would require a recalibration between the cameras.

Given that the average distance between the robot and the people it would be interacting with is expected to be beyond 60 cm, only the pupil reflection would be reliably detected; therefore, the gaze direction could be approximated to a vector perpendicular to the vector connecting both pupils in space. This would be a valid approximation of where the person is looking, as people unconsciously adjust their head orientation to align with their gaze in order to minimise eye muscle strain.

Figure 7.3 shows all the major processing stages for extracting the position of the eyes in the image space. Two images are taken by the eye tracker camera under different IR lighting conditions, one with the on-axis and one with the off-axis LED lights. This produces two images of the person's face where the one taken with the centre lights shows brighter pupils. Since the same camera is used to generate

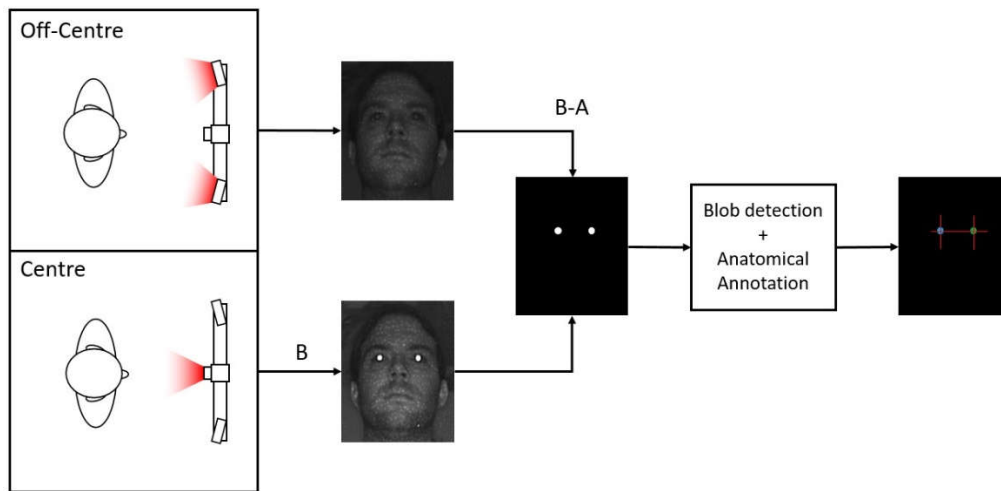


Figure 7.3 – Major stages of detecting the pupils using the IR camera setup. Images taken using different illumination sources are subtracted to reveal the position of the pupils. Subject of the images is a colleague that volunteered for the experiment

both images, an image subtraction generates a nearly completely black image with only the pupils and minor artefacts left. The image is then converted from grey scale to black-and-white, and each of the white blobs are detected through a blob-detector operation. The minor blobs are removed and the ones left are connected according to their likelihood of being two pupils next to each other (anatomical annotation).

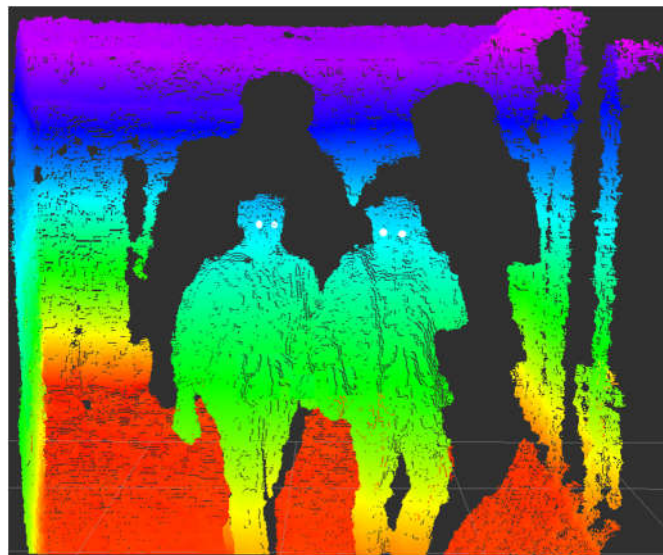


Figure 7.4 - 3D Point cloud capture by the Kinect of two subjects walking towards the robot. The pupils detected by the IR camera are projected onto the 3D point cloud in order to get a real-world depth estimate of where the eyes are in space.

Figure 7.4 shows the detected pupils from the IR camera onto a point cloud captured by the Kinect depth sensor. The colours represent the height of each point relative to the floor, and the dark grey areas

are not visible by the Kinect. Once the pupils are projected on the point cloud, the closes point in space is used to centre the pupil position, in this case represented by white spheres.

7.2.1 Control Board Design

A control board had to be created from scratch in order to drive the LEDs on the IR eye tracker in synchrony with the camera signals. The first components selected for the gaze detector were the camera, a Point Grey Flea ®3, a 1.3Mp (1328x1048pixels) USB 3.0 camera (Model FL3-U3-13S2M-CS) and the lenses (Fujifilm model DV3.4x3.8SA-SA1). The selected camera was equipped with a General Purpose Input/Output (GPIO) connector that provided logic level signals for the images that were taken, which is very useful to control the on- and off-axis illumination directly without software intervention.

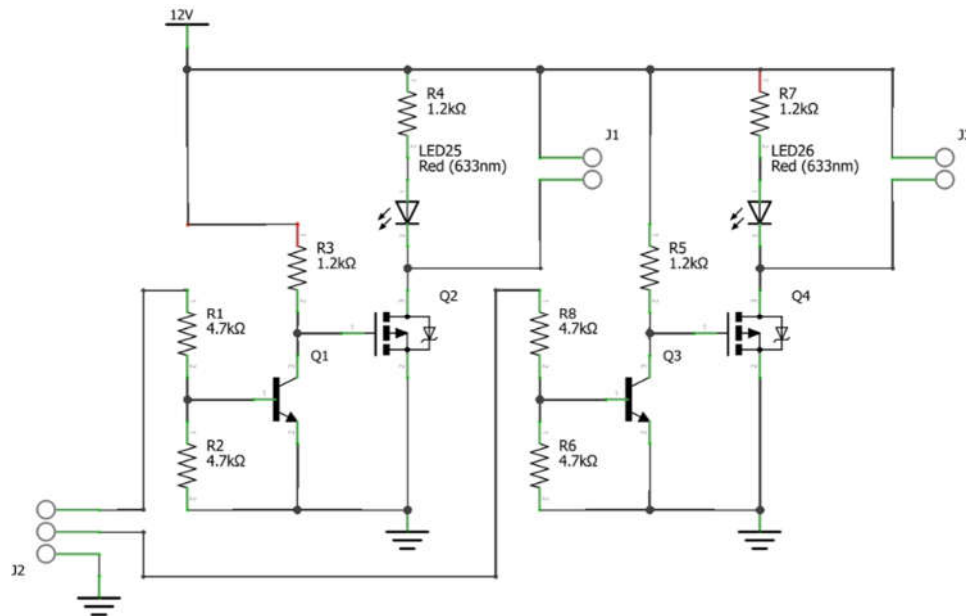


Figure 7.5 – Electronics circuit diagram for the control board, responsible for controlling both on- and off-axis light sources and interfacing with the camera

The selected lenses were equipped with zoom, focus and electronic shutter control. Zoom was set to its widest angle and focus to infinity, while the shutter was locked at a constant “open” position. Next, an IR filter Edmund Optics (model number: 43-949) was used to block all visible light, letting only the IR illuminated scene to be visible by the camera.

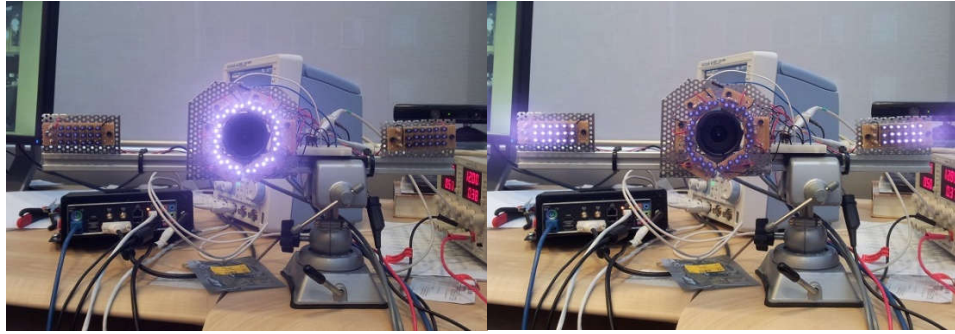
The circuit board is composed of two symmetrical LED controllers, one for each illumination source, as shown in Figure 7.5. The camera’s GPIO provides two square waves, 180 degrees apart,

synchronised with the half the capture rate of the sensor. This effectively provides two logic signals where their ON state represents alternating frames on the camera which are used to directly drive the LEDs with no software required. The GPIO cable from the camera is split into individual wires, which are connected to their respective terminals in the control board.

Each of the wires that carries a square wave signal are connected to terminal J2, with the on-axis signal in pin 1 and off-axis in pin 2. The signals do not drive the MOSFETs directly, which are responsible for the turning on the IR LEDs; instead, they control a transistor that acts as a buffer to ensure that only a minimal amount of current is being drained/sunk from/into the GPIO line. Notice that the visible light red LEDs on the control board were added for debugging purposes. The LEDs from the on- and off-axis sources were then attached to terminals J1 and J3, respectively.

7.2.2 Prototypes

Once the structure of the data collection and the electronics design were completed, a first prototype was needed to prove the validity of the idea. The number of LEDs required for the on- and off-axis light sources, as well as the distances between the LEDs and the camera, were still unknown, and therefore some adjustments would be necessary through prototype iterations. Figure 7.6 shows the first prototype with its different light sources illuminated one at time. The first prototype used 24 LEDs placed around the camera lenses for the on-axis illumination and 36 LEDs, 18 on each side, for the off-axis illumination. The goal was to make each light source illuminate the scene equally, with the only major difference being the pupil reflection on the on-axis source. However, the LEDs needed to be placed even closer to the camera's CMOS sensor in order to detect any noticeable differences on the pupils. Distances and angles of the off-axis light sources were determined empirically by comparing the illuminated scene from the on-axis with the one illuminated from different positions of the off-axis sources. The pictures of all three prototypes can be seen in Figure 7.7.

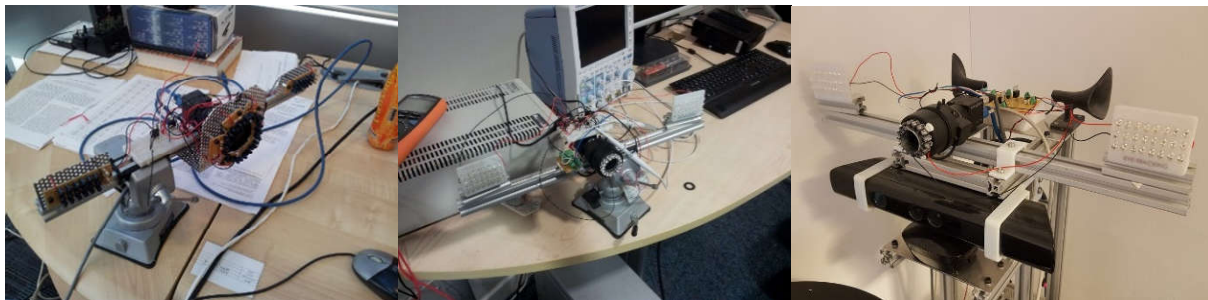


(a) On-axis LEDs illuminated

(b) Off-axis LEDs illuminated

Figure 7.6 – IR light from centre and off-centre IR LEDs captured by a mobile phone camera. CMOS sensors, like the ones in mobile phones, are able to capture both visible and infrared light.

The second prototype used 3D printed parts in order to move the on-axis LEDs inside of the camera lenses. This resulted in much brighter pupil reflection, however, it came with the cost of reducing the camera's view angle to around 35 degrees (± 17.5 degrees from the screen centre point) due to the 3D printed piece inside the lenses. A new 3D support piece for both off-axis light sources was also added.



(a) First prototype

(b) Second prototype

(c) Third prototype

Figure 7.7 – Different prototypes of the eye tracker. The first prototype was built in a day as proof of concept. The second and third one used 3D-printed parts and represented several minor interactions in design.

The third prototype retained the previous 3D-printed parts and added four new brackets in order to secure the Kinect sensor to the gaze detector's support beam, thus making it into one single structure. The last step in the development of the gaze tracker prototype was to register the Kinect's depth camera to the pupil detector's camera after mounting it on the robot. After initial tests with the moving robot, it was noticed that the calibration between cameras started to drift due to the vibration introduced by the omni wheels. After only a few short-distance runs, the cameras would move apart ever so slightly, thus rendering the recorded data useless for the sensory fusion. Despite the fastened brackets and screws, it would be necessary to create an entirely new single 3D-printed support piece that could secure both cameras firmly. Due to time constraints, however, it was decided to mount the eye tracker on an

older robot platform equipped with air filled rubber wheels, which acted as vibration dampeners for when the robot was moved around for data collection.

7.3 Depth-Based Gaze Estimation

The depth image provided by the Kinect sensor can be converted into a 3D point cloud with very little operations overhead, which opens the possibility for ample post-processing for detecting people. A common approach for detecting people in point clouds is splitting the cloud into clusters based on their points adjacency and classifying the clusters as people or not. However, since the eye tracker camera and the depth camera from the Kinect are registered, pupils detected in the IR gaze-detector image can be correlated directly to their respective 3D points in the depth image. This means that no anatomical-constraint-based search algorithms are necessary to find the position of the eyes in 3D space. Their position can also be used to find other areas on the face in the point cloud in order to better estimate one's gaze along the floor plane.

In addition to the proposed sensor-fusion method for combining gaze information, three other gaze estimation methods were implemented using only the anatomical features extracted from the point cloud sub-cluster representing the person's face. These other methods were used to explore different approaches that could be implemented in order to achieve more accurate results. All three extra methods are described in subsections 7.3.1 to 7.3.3 below.

7.3.1 Eyes Gaze Vector

The first gaze estimation method uses the corresponding 3D points on the point cloud from the 2D positions of the detected pupils in the IR image to calculate the gaze direction α , as shown in the top-view diagram in Figure 7.8. Once the pair of image blobs representing the pupils is identified, their centre is calculated and used to retrieve the corresponding depth from the depth image. The centre of the blob is often in between pixels, hence, the neighbouring pixels on the depth image are used to interpolate its position in space. Figure 7.8 depicts how the vectors and angle α are all related relative to the subject's face.

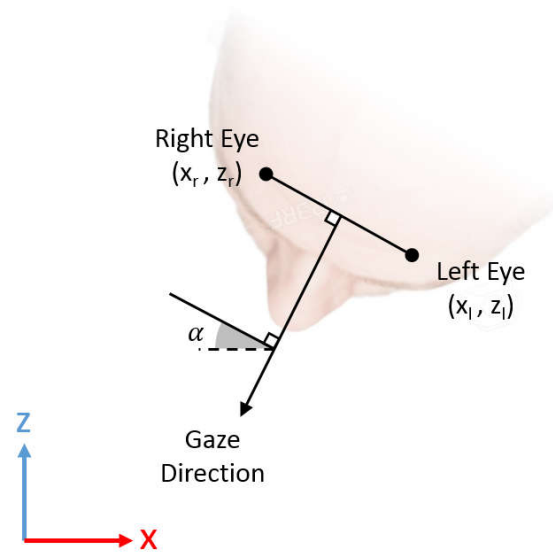


Figure 7.8 – Top-view diagram of gaze estimation using the 3D position of the left and right eyes and nose. The robot reference frame can be seen on the left, and angle α corresponds to the XZ-plane angle of the person's gaze.

The gaze direction of interest lies on the XZ plane, the floor plane, therefore the gaze angle can be calculated using Eq. (7.1).

$$\alpha = \tan^{-1} \left(\frac{x_l - x_r}{z_l - z_r} \right) \quad (7.1)$$

7.3.2 Nose Gaze Vector

The second method assumes that the person's head is facing forward, or at a small-pitch angle — as it would normally be when looking at someone of similar height. The first step for detecting the nose is detecting the eyes in space, which was covered in the previous subsection 7.3.1. Next, all points that fit within a vertical cylinder 6 cm in front of the eyes are tested to find the tip of the nose, as shown in Figure 7.9. The radius of the cylinder was set to 1 cm, the top was set at the eye height, and the bottom at -5 cm from the eye height. Assuming the vectors representing the eyes in space are $P_{left} = [x_l, y_l, z_l]^T$ and $P_{right} = [x_r, y_r, z_r]^T$, and the vector connecting the eyes is $\vec{E} = P_{left} - P_{right}$, the midpoint between the eyes can be found using $\vec{P}_{middle} = \vec{P}_{left} + 0.5\vec{E}$. Last, the centre of the cylinder is calculated using a dot product to find the normal vector between the eyes and placed it 6 cm along it from the middle point, $\vec{P}_{nose} = \vec{P}_{middle} + k[-\vec{E}_z, \vec{E}_y, \vec{E}_x]$, where $k = 0.06/\|\vec{E}\|$. The last stage is

finding the point with furthest distance from \vec{P}_{middle} among all points within the cylindrical volume representing the nose (i.e. the nose tip point P).

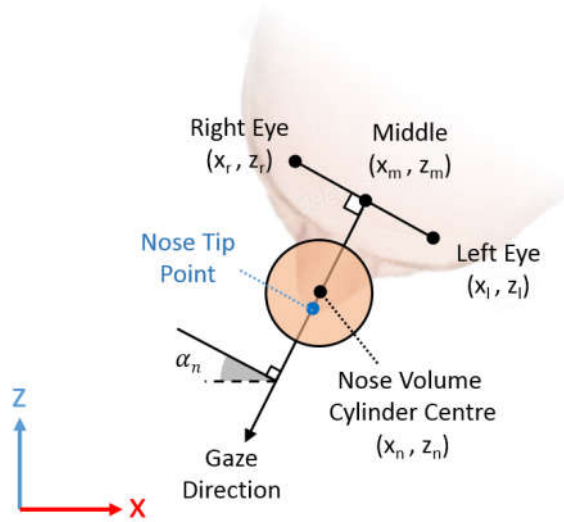


Figure 7.9 – Top-view diagram of gaze estimation using left and right eyes and nose. The robot reference frame can be seen on the left, and angle α corresponds to the XZ-plane angle of the person's gaze.

The final nose vector can be calculated by the angle of the vector connecting the middle points to the nose tip point:

$$\alpha = \tan^{-1} \left(\frac{x_t - x_m}{z_t - z_m} \right) \quad (7.2)$$

7.3.3 Forehead Gaze Vector

The third gaze estimation method is a variation of the second method where the gaze angle corresponds to the angle of a plane fitted on the 3D points representing the person's forehead. The points are selected using the same volume intersection method as in the second method, however, the cylinder is moved to where the forehead would be, given the known position of the eyes.

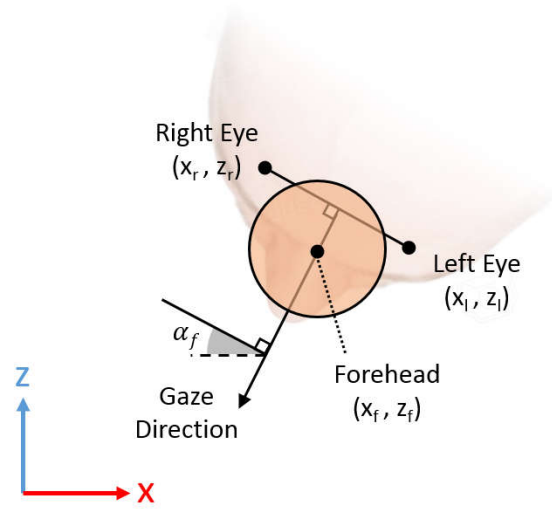


Figure 7.10 – Top-view diagram of gaze estimation using left and right eyes and forehead. The robot reference frame can be seen on the left, and angle α corresponds to the XZ-plane angle of the person's gaze.

To fit a plane on the 3D points that lie within the cylinder, a linear fitting was done, which in this case was done by calculating the principal components of the plane. The perpendicular component is the least important component of the three in terms of contribution, but it is the one component that denotes where the person's face is pointed. First, the mean of all points is calculated:

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} = \frac{1}{n} \sum_{i=1}^n \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad \text{for } n \text{ points}$$

Each coordinate of each point is then subtracted by its respective mean, resulting in a zero mean $n \times 3$ matrix P :

$$P = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ \vdots & \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} & z_n - \bar{z} \end{bmatrix} \quad \text{for } n \text{ points}$$

Next, the 3×3 covariance matrix C is calculated using the new zero mean matrix P :

$$C = P^T P$$

Finally, the eigenvalues (λ_j) and eigenvectors (p_j) of the covariance matrix are calculated:

$$(C - \lambda_j I)p_j = 0 \quad \text{for } j = 1, \dots, 3$$

The smallest Eigen vector is then the perpendicular vector, from which we can calculate the angle α_f .

7.4 Proposed Sensor Fusion Gaze Estimation

The IR camera for the gaze detector and the Kinect sensor are complementary in terms of spatial sensing. While the IR camera has a higher resolution of 1328x1048 pixels, it lacks the depth estimation. On the other hand, the Kinect sensor has a lower resolution of 640x480 pixels, and it has an approximate 12-cm depth variance error at 2.5 m, makes determining the position of the eyes for long range at longer ranges unreliable.

In order to achieve a more accurate gaze angle estimation than the one achieved by either method independently, the higher image resolution of the pupil detection was combined with the estimated depth of the pupils to more accurately estimate the gaze angle. Figure 7.11 shows the diagram of the data provided from both sensors.

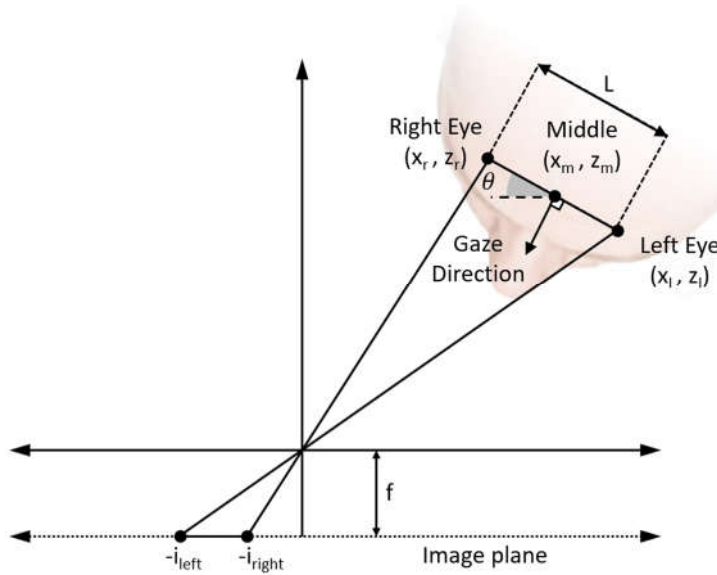


Figure 7.11 – Diagram of how the position of the eyes in space are projected on the IR camera's image plane

The projection of the pupils on the IR camera's image plane is shown as i_{left} and i_{right} , and can be calculated using the equations presented below [144]:

$$i_{left} = -\frac{x_l}{z_l}f \quad i_{right} = -\frac{x_r}{z_r}f \quad (7.3)$$

In order to reduce the effects of the noisy depth measurements of the Kinect sensor, the middle point between both eyes is used as the relative reference for the gaze in space. First, the middle point is

calculated, as demonstrated in subsection 7.3.2; next the ‘filtered’ position of the eyes in space is calculated using the middle point, finally, the position of the pupils is estimated using the equations in (7.4) [144], where L is the fixed distance between pupils in the real world:

$$\begin{aligned} x_l &= x_m + \frac{L}{2} \cos(\theta) & x_r &= x_m - \frac{L}{2} \cos(\theta) \\ z_l &= z_m - \frac{L}{2} \sin(\theta) & z_r &= z_m + \frac{L}{2} \sin(\theta) \end{aligned} \quad (7.4)$$

If equations in (7.4) are used to replace their respective terms in (7.3) [144], the distance between both pupils on the IR image plane can be written as:

$$i_{left} - i_{right} = \frac{x_m - \frac{L}{2} \cos(\theta)}{z_m + \frac{L}{2} \sin(\theta)} f - \frac{x_m + \frac{L}{2} \cos(\theta)}{z_m - \frac{L}{2} \sin(\theta)} f = 4fL \frac{z_m \cos(\theta) + x_m \sin(\theta)}{L^2 \sin(\theta)^2 - 4z_m^2} \quad (7.5)$$

The distance between pupils, or interpupillary distance, L , may vary from person to person, which could affect the final results; however, studies have determined that an average interpupillary distance for an average adult is 67 mm [145].

With the interpupillary distance as a constant, the only variable left is the gaze angle θ . Since the real distance between the eyes is much smaller than the distance from the eyes to the camera, or $L \ll z_m$, Eq. (7.5) [144] can be further simplified. In addition, the equivalence term $D_{pupils} = i_{left} - i_{right}$, was introduced for convenience, hence:

$$D_{pupils} = \frac{x_m - \frac{L}{2} \cos(\theta) - x_m - \frac{L}{2} \cos(\theta)}{z_m} f \quad (7.6)$$

Finally, the gaze angle from the sensory function can be calculated using Eq. (7.7) [144].

$$\theta = \cos^{-1} \left(\frac{D_{pupils} z_m}{fL} \right) \quad (7.7)$$

Notice that the distance of the subject from the centre of the camera, x_m , is not necessary when calculating the gaze angle as per Eq. (7.7) [144] in practice; however, the closer to the centre of the camera the person is, the more accurate the estimated angle is calculated, and whenever $x_m > 0.5$, a valid gaze angle cannot be calculated. This effect was noticed as $D_{pupils} z_m \ll fL$, which estimates a near 0 corrected gaze angle in all cases.

Since the analytical solution for the angle was not reliable, a non-linear solution was used. The Newton-Rhapson method was selected for its fast convergence in finding the roots of real function. Eq. (7.5) was then re-written as a function of θ [144]:

$$f(\theta) = \frac{z_m \cos(\theta) + x_m \sin(\theta)}{L^2 \sin(\theta)^2 - 4z_m^2} - \frac{D_{pupils}}{4fL} \quad (7.8)$$

And its derivative in respect to θ [144]:

$$\begin{aligned} \frac{d(f(\theta))}{d\theta} = & 4fL^3 z_m \frac{\sin(\theta)^3 - 2 \sin(\theta)}{(\sin(\theta) L - 2z_m)^2 (\sin(\theta) L + 2z_m)^2} \\ & + 4fL \frac{-\cos(\theta) \sin(\theta)^2 x_m L^2 + 4 \sin(\theta) z_m^3 - 4 \cos(\theta) x_m z_m^2}{(\sin(\theta) L - 2z_m)^2 (\sin(\theta) L + 2z_m)^2} \end{aligned} \quad (7.9)$$

A visual representation of function $f(\theta)$ is shown in Figure 7.1. Real data was collected to serve as valid parameters for the function plotting, and a top view representation of the person in each scenario shows their relative position and orientation from the camera. All plots are limited to +/-40 degrees because that is the actual horizontal range of the IR camera after the on-axis LED ring was installed. The plots show that when the person is facing the camera, $f(\theta)$ only has one root, whereas when facing any other angle $f(\theta)$ has two roots. This is true for any detectable gaze within the camera's horizontal range, as verified by empirical testing. When facing the camera directly, the single root of $f(\theta)$ is the actual estimated angle; however, when facing anywhere else, $f(\theta)$ returns two roots: one that corresponds to the actual angle of the gaze while the other is either its mirrored value, if the person is standing at the centre of the view, or a shifted value, when the person is standing elsewhere.

The Newton-Raphson method converges to the nearest local minimum to the starting point of the search derivative function (7.9), hence it is unable to return all the roots of the function (7.8). In order to return both roots, two searches are performed from different starting points, namely -40 and 40 degrees. If both searches converge on the same point, it is assumed only one root exists and that is the final angle. If both searches diverge, the last stage is determining which root corresponds to the real angle, which can be done by using an estimated angle from another method. In this implemented

solution, the gaze angle estimated by the forehead vector is compared to both angles found via the Newton-Raphson, and the closest angle is the one selected at the end.

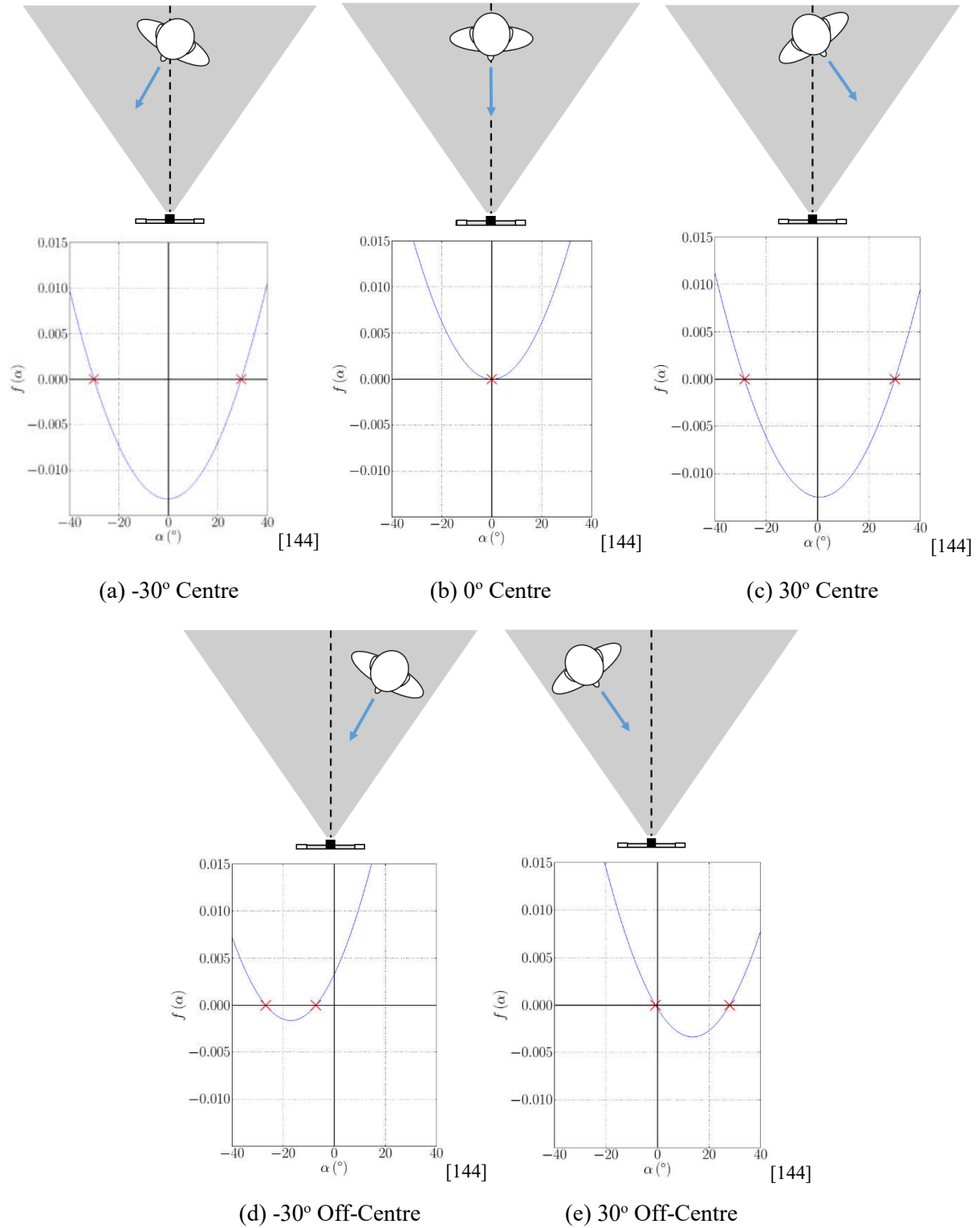


Figure 7.12 – Gaze angle estimation at different positions in front of the gaze detector. Depth and eye positions used to generate graphs were collected from a real volunteer.

7.5 Experiments and Results

The gaze tracker was mounted on the new assistive robot for the experiments, however, after a few initial runs, the vibration created by the omni wheels caused the Kinect and the eye tracker camera to move on the frame. The effect of the vibration was to the degree that it was impactful on the initial readings. As such, it became necessary to re-register the cameras for almost every new experiment, rendering previous datasets invalid for subsequent camera registrations. This introduced a practical obstacle: namely, that the experiment meant to capture valid data as a consequence of the robot moving was not possible.

It was necessary to mount the gaze tracker on a previous assistive robot prototype equipped with rubber wheels, previously shown in Figure 3.1 in Chapter 3, in order to minimise vibration on the gaze tracker frame. This allowed experiments to be performed where the robot had to move towards the people. Two sets of experiments were designed to evaluate the accuracy of the gaze-estimation methods made reference to in Sections 7.3 and 7.4 above. The first set consisted of keeping the robot still while the subject stood at 1.5 m, 2 m and 2.5 m away from it whilst facing nine markers placed on the wall so that the subject's face angle relative to the robot would be at intervals which ranged from -40° to $+40^\circ$, given 10° increments. Figure 7.13 shows a top view of the relative position of the robot and the subject during the experiments.

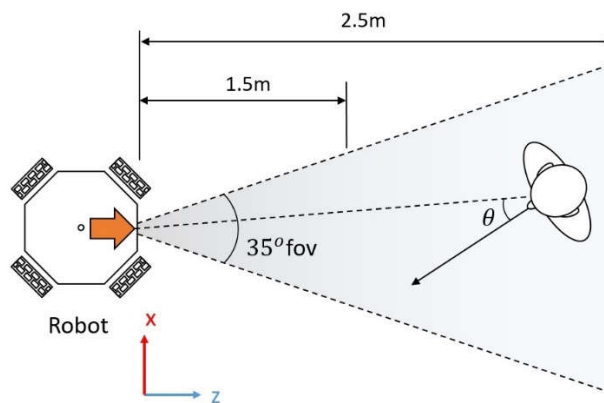
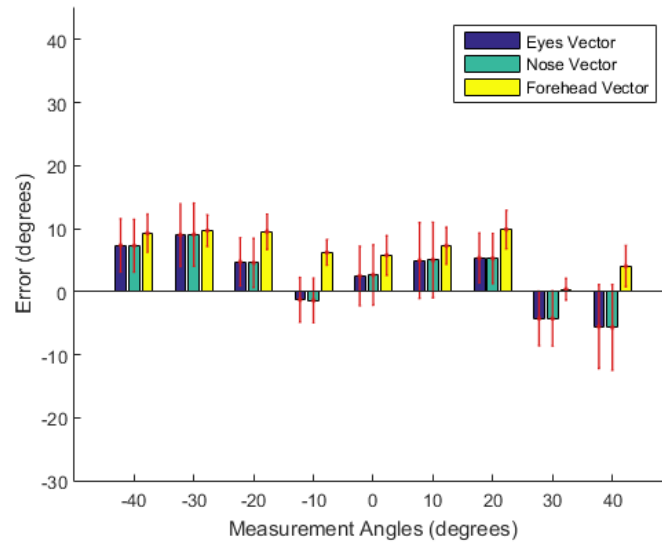


Figure 7.13 – Diagram of how the experiments were conducted. Subjects would stand at distances of 1.5, 2.0 and 2.5 m from the robot and look at wall-markers to ensure ground-truth gaze direction.

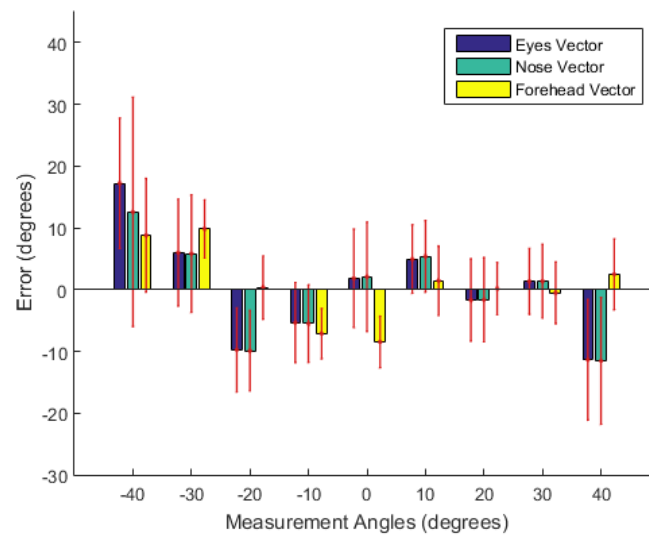
The second set of experiments consisted of creating scenarios in which the subject walked toward the markers whilst his gaze was held at a pre-determined marker. This set of experiments was intended to ascertain the extent to which the motion of the subject might interfere with the accuracy of the robot's gaze-estimation capabilities.

7.5.1 Experiments for Depth-Based Gaze-Estimation

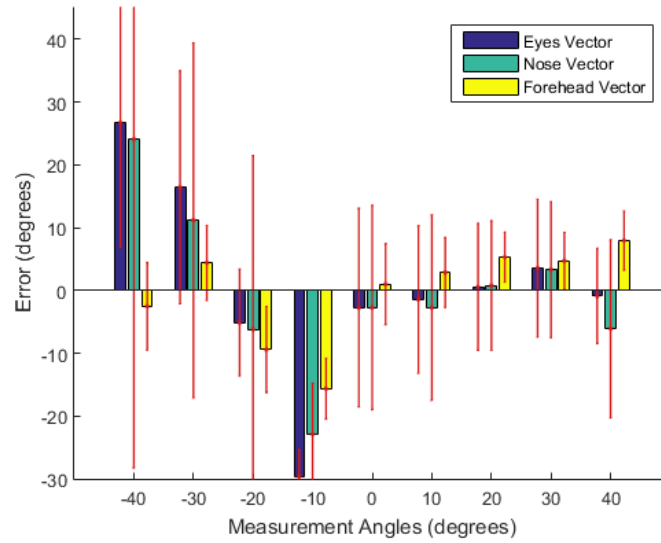
The results from the three experiments in which the robot maintained a stationary position indicate that as the subject moves away from the robot, the accuracy of the gaze-estimation decreases. The bar-graphs shown in Figure 7.14 show the error of the depth-based gaze estimation methods in degrees for each of the nine gaze directions in each of the three measured distances.



(a) 1.5 m



(b) 2.0 m



(c) 2.5 m

Figure 7.14 – Results from static experiments. Overall errors for each distance indicate that the accuracy is inversely proportional to the distance.

The height of the bars indicates the mean error of the session and the red lines indicate the standard deviation of the error. It is clear that as the person moves away from the robot, the overall mean error and standard deviation error increases for all measured gaze angles. The forehead gaze estimations present an overall lower mean error compared to the eye- and nose-vector methods throughout the experiments.

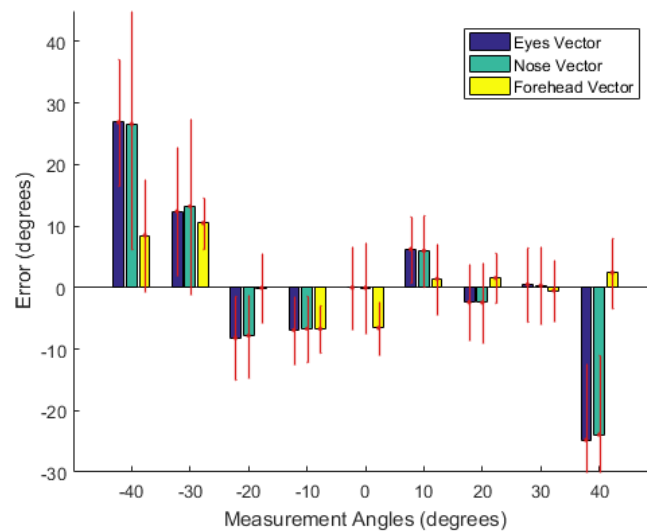


Figure 7.15 – Average performance of the gaze detector when subjects moved towards the robot. The forehead vector estimation presented the lowest error margin compared to the eyes and nose vectors

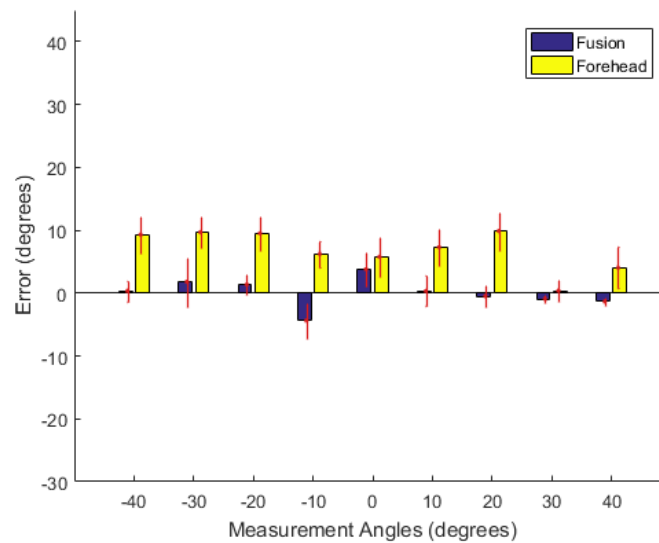
This is due to the number of points used to approximate the gaze vector from the forehead, compared to two points for the eyes on the other methods. It is also possible to notice a skewed error

accumulation on the negative angles which is expected due to the displacement of the dot-projector from the Kinect. Using a time-of-flight camera could reduce the skewed error, but it would also interfere with the eye IR tracking, as the time-of-flight requires its own IR illumination source.

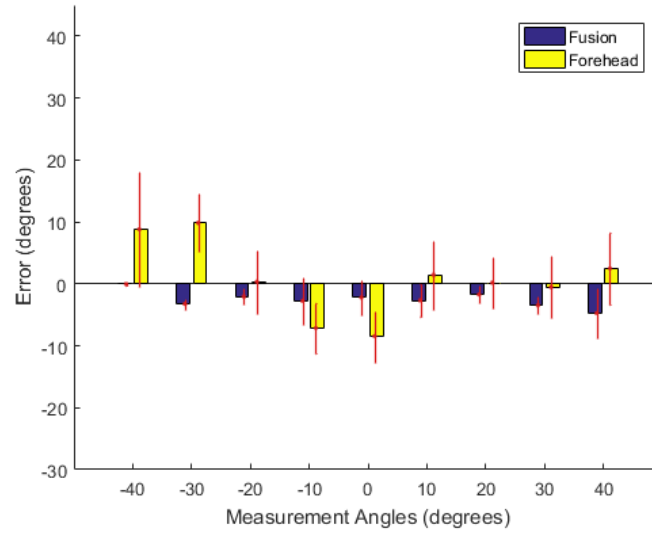
The moving-robot experiment was averaged over the three distances and the results can be seen in Figure 7.15. In practice, the moving robot and moving person scenario demonstrated similar results, with the only difference being the higher noise caused by the vibration of the robot's chassis while moving

7.5.2 Sensor Fusion Results

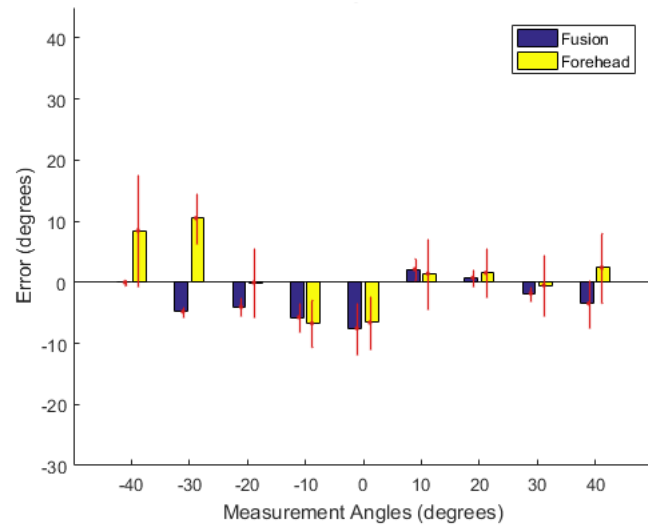
Using the sensor fusion method presented in sub-section 7.4, the IR image with the detected pair of eyes was combined with the depth information for all sets of experiments recorded. For comparison, the forehead-vector gaze estimation method, the most accurate of the three previous methods, was added to the results.



(a) 1.5 m



(b) 2.0 m



(c) 2.5 m

Figure 7.16 – Results of the sensor fusion method compared to the forehead vector. The proposed sensor fusion method consistently demonstrated higher accuracy in estimating the angle of gaze

The results in Figure 7.16 show the mean error as bars and standard deviation as red vertical lines for each of the three distances and all nine angles tested. The sensor fusion method demonstrated an overall lower mean error compared to the forehead vector, with errors smaller than 5 degrees for up to 2 meters. For the scenario where the person is moving towards the robot, the fusion method still outperformed the forehead vector, as shown in Figure 7.17.

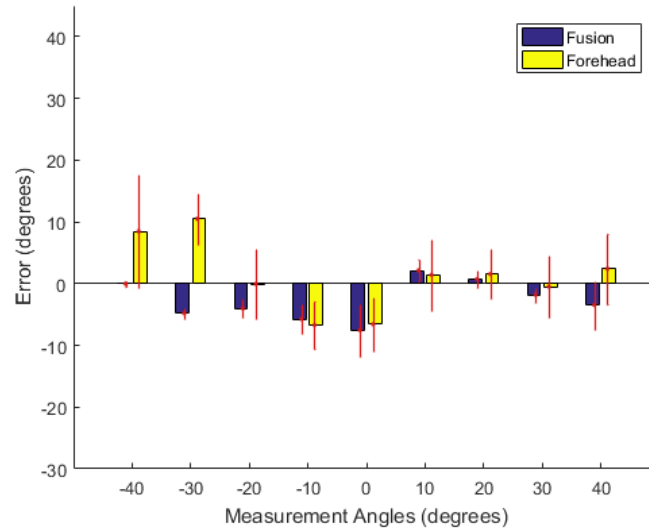


Figure 7.17 – Sensory fusion results for a moving scenario compared to the forehead vector estimation method. The proposed sensor fusion method still outperforms the forehead vector estimation in a moving scenario

7.5.3 Real-World Evaluation

In the previous subsections, 7.5.1 and 7.5.2, the accuracy of the gaze tracker was measured under specific test conditions, where both the distance and orientation of the people were known; however, testing the gaze tracker under un-scripted conditions was necessary to ensure its effectiveness in the wild. Without reference cards to look at, the direction of the person’s gaze within the field-of-view had to be compared against the general orientation of their faces. In addition, multiple people were included in these scenarios.

The originally intended real-world evaluation was that of using the gaze tracker on a moving robot, however, the vibration created by the omni wheels caused the cameras to drift out of registration within a few minutes. In lieu of the intended evaluation, several scenarios were recorded with people talking and walking in proximity of the robot, while their gaze directions were detected in real time and later assessed for correctness using the general direction of the person’s face compared to the estimated direction of the gaze. Three of the many scenarios recorded are shown in the images below. In each scenario, the estimated angle of the gaze was filtered using a Kalman filter to reduce the jittering resulting from the noise introduced by the IR and depth-sensor cameras.

In the first example (Figure 7.18 below), three people were within 2 m of the robot and chatting amongst themselves while casually gesturing at the robot. This scenario was designed to demonstrate

how the robot would perceive people who were purposely trying to interact with it in a real situation. The robot can detect the direction of the gaze of persons B and C, shown as red arrows, as both their pairs of eyes are visible in Image (a). Person C is located within 1.5 meters of the robot and looking at the general direction of the robot whilst waving his hand at the robot. This would represent a situation whereby an individual can choose to initiate an interaction with the robot, in this case, by simultaneously gesticulating and looking at it.

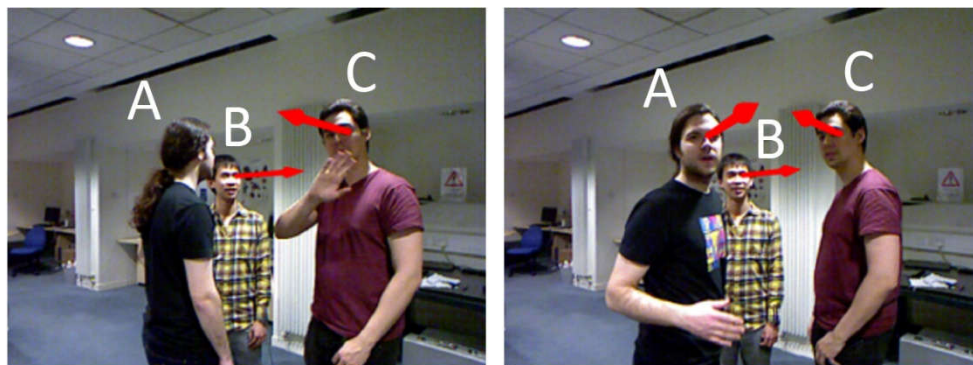


Figure 7.18 – Example of three subjects within 1.5 m of the robot. Gaze is detected using sensory fusion and processed through a Kalman Filter to reduce jitter.

The second example (Figure 7.19 below) depicts a static environment in which three people are talking within 2.5 m of the robot. In Images (a) and (b), the red arrow represents gaze-direction estimated for person B, as both his eyes are visible to the IR eye-tracker. The gaze of persons A and C are not detected in this scenario, as pairs of eyes must be fully visible to validate the detected gaze. During the recording of the scenarios, the gaze-direction of person B was consistently detected as he diverted his gaze from person A to C, and vice-versa, throughout the conversation.

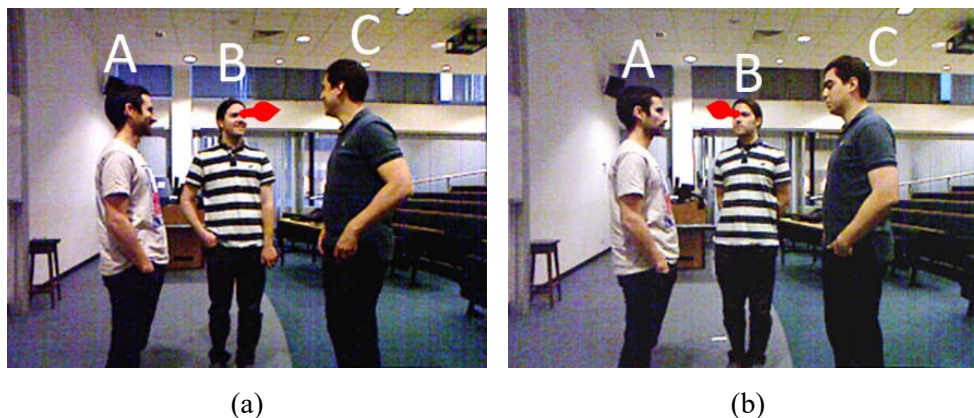


Figure 7.19 – Static scenario with three people chatting near the robot. The gaze of persons A and C is not detected since both eyes must be visible to the gaze tracker

In the third example (Figure 7.20 below), three individuals are walking in front of the robot. Persons A and B are chatting whilst walking side-by-side, and person C is walking by himself, momentarily stopping to look around from time to time. In Image (b), person B is seen looking at the robot directly, which causes the red arrow to resemble two circles due to perspective projection. Person C's gaze is not detectable as he is not looking in the vicinity of the robot, and person B's gaze cannot be detected as only one eye is clearly visible.

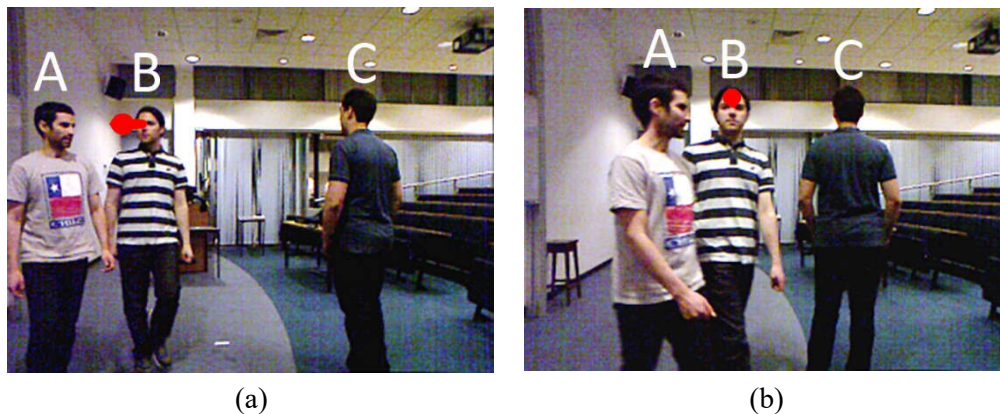


Figure 7.20 – Dynamic scenario with three people. Persons A and B walk by whilst engaged in conversation, and person C is walking by himself.

These three examples illustrate some of the commonly found scenarios in most populated environments. The direction of the arrow superimposed over the images closely approximates the direction of gaze of the person, judging by the images. In a future version of the assistive robot, HRI could be initiated at the moment a person looks at the robot, even during a conversation with other people, by briefly looking at the robot and gesticulating. The images featured in Figure 7.18 to Figure 7.20 serve to demonstrate that the gaze tracker is already capable of detecting when such an interaction is likely to occur.

Lastly, a comparatively unstructured scenario was devised in which random people would simply approach the robot at their normal speed eventually passing by it, without prior awareness of its presence. The purpose of devising this scenario was to evaluate the degree to which varying the walking speed relative to the robot's might affect the gaze tracker. Figure 7.21 below illustrates the recorded scenario where people were leaving a lecture hall. People are seen looking directly at the robot due to

their own curiosity, which presents an optimal condition in which to evaluate how closely the robot can detect gazes that may be intended to initiate an interaction.



Figure 7.21 – Dynamic scenario in which people are walking along a corridor whilst the robot remains stationary. The arrows represent people’s estimated gaze-direction up to 2.5 metres from the robot.

It is, however, noticeable that both the position and orientation of the red arrows in this scenario are not as closely correlated to those of the person’s own eye position and gaze direction, respectively. Prior to the recording of this scenario, the robot had to travel across the building to get to the lecture hall, which may have added a small drift in the registration between the cameras. In addition, the error in the angle effect is partially explained by the relatively low 30Hz refresh rate of the depth camera. At 30Hz, the electronic shutter of the CMOS sensor inside the depth camera cannot capture a frozen instance of the movement, but rather a small portion of the movement itself is captured, thus causing a blurring effect. This blurring effect on the depth camera manifests itself as a larger volume representing travel of the moving object between frames. As a result, the depth estimation position is skewed and the gaze angle is not as accurate compared to slow moving or stationary persons.

7.6 Conclusions

A detailed development of the long-range IR eye tracker was presented in this chapter, from initial concepts, hardware schematics, prototypes and improvements on gaze detection accuracy via sensory

fusion. The motivation for designing the long-range gaze tracker was to allow the robot to determine whether people are looking at it from a distance, in order to be able to pro-actively approach people and/or initiate HRI.

The IR tracker was built using a Kinect, an IR camera and two sets of IR LEDs to illuminate the subject's face. A custom driver board was made to synchronise the IR camera frame signal with the IR LED illumination so that alternating frames were captured with centre and off-centre IR light exposure. By subtracting the two images from the alternating frames, only the pupils were detected in the IR camera high resolution image. After processing the final image for blob-detection and anatomical feature association, the position of the pupils could be determined.

The IR gaze tracker is capable of detecting people's gaze direction at 30Hz up to 2.5 meters away, with mean errors less than 10 degrees and a standard deviation of 5 degrees. This tracker was capable of detecting angles ranging from -40° to $+40^{\circ}$ at different positions ranging from 1.5 m to 2.5 m. Another important feature of this tracker is that no calibration is required.

In order to achieve higher accuracy, the higher image resolution of the IR camera and the depth position from the Kinect-generated point cloud was combined through an analytical solution for pupil detection given the distance from the camera and their pupil distance on the image. The angle of the estimated gaze approximated via solving for the roots of the analytical non-linear equation using the Newton-Raphson method.

Future improvements on the IR gaze tracker would consist of designing a new frame for both cameras and IR illumination that should prevent cameras from drifting due to vibration coming from the omni wheels while in motion. A current limitation of the gaze tracker is also its narrow field of view (about 35 degrees). The centre IR LEDs had to be placed inside the camera's lens housing to achieve the desired pupil reflection effect. A future version of this system could also incorporate reflective housing for the LEDs around the camera in order to focus the light and allow more free space for the camera's sensor.

Chapter 8

8 Conclusion

In the United Kingdom alone, an ageing population is cared for by a decreasing number of assistive staff in hospitals and general practices. This phenomenon, projected to continue, is one of the main driving forces of the advancement of an array of assistive technologies. At the present time, the field of assistive robotics nonetheless requires a number of years of development before it becomes a widely adopted technology; one which, it is hoped, will someday be as ubiquitous as mobile phones are at the present.

This research sets out to provide some form of guidance with regard to the development of future mobile assistive robots, while presenting plausible, relevant and market-aware solutions for common problems which are found in each of the main development areas explored throughout the research. The hardware solution itself is comprised of an autonomous platform, made out of commonly found components which are easily replicable.

The inextricably-linked software solutions which operationalised the various tasks performed by the hardware were designed in context of the Robot Operational System (ROS). This approach serves to maintain an inherent openness to integration with a number of other software modules designed and developed by other institutions. Such an open-source approach toward software development ensures that innovation is likely from as many contributors as possible, thus accelerating the continuous development of synergistic hardware-and-software solutions.

This thesis thus presents a holistic and sufficiently detailed design, in which navigation, path-planning, floor-classification and gaze-detection were all operational elements of the mobile assistive robot which is the protagonist of this research programme. The contribution to knowledge of this thesis has been to plan, design, and create a working prototype on which the development of future

autonomous assistive robots for indoor dynamic environments may be based. The task of designing an operational prototype from the initial planning stages involves detailed enquiry and breadth of analysis aspects of mechanical, electrical, electronics and software domains.

Each chapter of this thesis is thus envisaged as a layer in a progressively hierarchical structure, one which serves to demonstrate how and why an array of unrelated systems can be synergistically integrated so as to result in a working prototype of an autonomous mobile assistive robot. Firstly, a working prototype was designed and constructed to allow holonomic movement, to minimise vibration and to enable both 2D and 3D sensing.

The eventual omni-wheel solution chosen for the prototype generated a level of vibration which could then be used to identify various floor-surfaces on which the robot would typically traverse. Information about in-situ floor-surface variation led to hypotheses about challenges likely to be of concern in typical medical-care settings. An example of this would be the likely need to adjust the PID parameters on the motor controller in order to reduce slippage when moving from carpet to wood. This information could also be used to add floor information to various preliminary mappings of an array of static and dynamic environments.

During the mapping of the environment, a leg-detection method was used to locate people in the environment. Such people, over intervals of time, were tracked in terms of their unpredictable direction and speed of travel. This permitted the robot to make more accurate assessments of where to position itself in order to avert collision intervals. Eye-tracking capabilities would enhance the ability to make accurate assessments of unpredictable direction and speed of travel, in that a person's gaze direction, it was theorised, could be combined with location-detection data, and thus heighten accuracy in terms of predicting a given person's likely direction of travel.

Throughout the entirety of the robot prototype's development, several problems spanning multiple disciplines were thus addressed. There were, however, a number of areas of enquiry which will be the subject of future research. Firstly, the leg-detection algorithm, introduced in the pre-processing phase of the dynamic mapping, presented a higher amount of false positives when assigning legs of randomly-

distributed groups of people. A possible solution might be to introduce depth and/or colour features into the classification step. Secondly, the dynamic path-planning algorithm uses a linear extrapolation of each person's current trajectory in order to predict their future position.

By introducing a data-based Bayesian model of human path-planning, it is theoretically possible to reduce several extra trajectory re-planning calculations triggered whenever a tracked individual started to move in a non-linear trajectory. Lastly, the IR eye tracker has a narrow field of view of approximately 35 degrees; this is due to the required centre-based LED illumination which needs to be positioned as closely as possible to the camera. Further research could be carried out on different types of cameras, so as to allow for wider and shorter telescopic lenses to be used. In addition, using two IR cameras with different zoom levels could allow for both a coarse and fine gaze-direction estimation.

At the time of publishing this document, the field of mobile assistive robotics in hospitals is still an emergent one, and it faces both great technical and social challenges. On the technical side, the ever-changing scenario of fully-functioning hospitals; with patients of various levels of cognition and physical aptitude, busy schedules, and a multitude of physical layouts; proves a great challenge to the field in terms of both sensing and modelling tasks on the part of the robot agent. On the social side, the introduction of robotic assistants in hospitals is presently construed as a means of demonstrating future technologies, rather than a concrete solution for current problems such as under-staffed hospital facilities.

8.1 Technical Contributions of Thesis

Chapters 1 and 2 offered the reader an introduction and overview of development of the last most relevant milestones in assistive robotic technology, along with a core research in several disciplines used in today's mobile assistive robots. These chapters set the scene for a contextual understanding of the various technical contributions of the thesis, each of which is presented in turn as follows:

Chapter 3 presented all the major steps in the hardware development of the new assistive robot. The final design incorporated an omni-directional base with a passive swivel suspension capable of

maintaining all four wheels in contact with the floor at all times. Other assistive robots are equipped with a non-holonomic differential drive which limits their manoeuvrability in crowded environments.

The assistive robot does not require external power supplies or charges as all components are fit inside, which makes it an idea platform for testing and development. Two Lidars were added to the provided 360-degree coverage of obstacles and walls around it. Depth sensors were later added for future object and people recognition.

Chapter 4 presented a new method for identifying types of surfaces based on the chassis vibration generated by the robot while driving. Unlike other surface classification methods that rely on vibration generated on rubber wheels, this method developed is capable of processing the noisy vibration signal created by the rolling of the omni wheels. Four different floors commonly found in hospitals were used for training the classifier: granite, wood, plastic (PVC) and carpet.

Different robot speeds and manoeuvres were used to create the wide range of training data. The features extracted from the two-second window sample of the vibration consisted of 523 statistical and Power Spectrum Based features, which were then reduced to 11 features through PCA projection. Several classifiers were compared and the Extreme Learning Machine was able to classify the four different types of floors, regardless of the speed and manoeuvre used, with a minimum 85% of accuracy.

Chapter 5 presented a new method for mapping dynamic environments by combining leg detection and SLAM. The core mapping algorithm follows the HectorSLAM method, where each scan is aligned with the current map in order to keep track of odometry, and the map is updated whenever the robot moves or rotates above a certain threshold. The alignment uses the Gauss-Newton non-linear approximation method and multiple grid levels to increase resilience to large steps.

In order to avoid adding people's legs to the map during the mapping phase, leg detection was used to separate clusters of points that represented legs and the rest of the environment. All clusters not classified as legs were used to update the map. The final experiment consisted of running the robot in a

large crowded environment, where the proposed method generated a more accurate version of the map compared to the original HectorSLAM.

Chapter 6 presented to a minor contribution to localisation and a major contribution to path planning in dynamic environments. The first contribution is a way for combining the scan alignment sub-routine from HectorSLAM into a finite state machine that uses AMCL to locate the robot in case it is lost and switches it over to the scan-alignment system when the robot is certain of its location. This method uses a fraction of the processing power compared to particle-based methods and it doesn't require any odometry information, which saves space in the robot's internal messaging system.

The second contribution of this chapter is Dynamic Safe Interval Path Planning (DSIPP), an improvement over the Anytime SIPP. The key feature of the proposed method removes the pre-processing collision detection stage using discretised cells to represent the map in both time and space. The new method dismisses any memory pre-allocation and, instead, generates direct time intervals from checking collisions with every visible obstacle at every cell evaluated by the Weighted A* algorithm.

The collision detection is achieved by a simple circle-to-line intersection, which is both computationally inexpensive and provides a continuous value for the boundaries of the time interval. Several experimental simulations were run, varying the number of visible moving obstacles and the length of the time horizon; the proposed DSIPP demonstrated to be between 1.5 times (at 5 sec time horizon) to 4.5 times (at 25 sec time horizon) faster than the previous state-of-the-art solution.

Chapter 7 proposed a new method for combining eye tracking and depth sensing estimation to achieve a more accurate gaze estimation. The IR tracker was built using a Kinect, an IR camera and two sets of IR LEDs to illuminate the subject's face.

A custom driver board was made to synchronise the IR camera frame signal with the IR LED illumination so that alternating frames were captured with centre and off-centre IR light exposure. By subtracting the two images from the alternating frames, only the pupils were detected in the IR camera high resolution image. The pupil position was then fused with the depth estimation of the eyes from the

points cloud to achieve higher gaze accuracy at distances at to 2.5m away from the robot. This gaze tracker was tested in both static and dynamic scenarios and was able to estimate gaze from -40° to $+40^\circ$ with mean errors of less than 10° .

8.2 Future Work

This thesis has presented a complete design process of an open assistive robot platform to be used in hospitals and other indoor care centres. Several improvements to current state-of-the-art techniques from different areas were proposed, but more work can still be done to produce more robust solutions.

The hardware design of the robot was conceived to allow for a plenty of space for future development, by adding new sensors or new structures to the aluminium chassis. The addition of gas and temperature sensors, even a thermal camera, would allow the robot to perform periodic checks on the environment's condition during its patrol and present a report of any variations in temperature in the rooms or undesirable concentration of gases in any particular area. One particular sensor that would benefit the robot would be a Velodyne 360-degree 3D Lidar. The addition of this Lidar would remove the need for the two existing Lidars and allow a much more complete view of the environment, especially of objects below the Lidars that cannot be seen during the mapping phase.

During experiments when the prototype robot had to run on extra thick carpets, the additional drag created under the rollers made it difficult for the robot to move for prolonged periods of time without overheating its motor drivers, as more electrical current was constantly required to break the friction force. In hindsight, the gear ratio of the motors' gearboxes should be reduced to increase the maximum torque generated. This would reduce the top speed, however, during all experiments conducted where the robot had to move along with a person, 65% of its top speed was more than enough to perform all tasks efficiently; therefore, no changes in speed would be noticeable during its normal operation.

The proposed method for mapping dynamic environments only accounted for detecting legs, which demonstrates the effectiveness of the solution, but it could be expanded to detect other shapes. the original leg detection method proposed by [136] was also trained to detect wheelchairs and walkers,

none of which were available during the training phase and were not included in the final results. Future work could include detecting a variety of assistive walking devices as well as trolleys.

In this thesis, all experiments were conducted in a controlled laboratory setting. Further research needs to investigate the robot's action and ability to navigating around autonomously in a hospital environment. With a user interface and scheduling system in place, the robot could be deployed in a real hospital to be tested by actual staff. One possible interface method would be via the use of mobile phone apps or even adding a touch screen to replace the current monitor on the robot. In addition, adding a proper cover would both improve its appearance to the patients and protect its internal components against objects or liquids being dropped on them.

It is for certain that the use of mobile assistive robotics in clinics and home environment will be on the rise. As mentioned in a recent article of Science Robotics [146] in terms of the future grand challenges of robotics, social interaction that understands human social dynamics and moral norms represent a major research direction. This requires the development of robotic technologies that can be fully integrated with our social life, showing empathy and natural social behaviours. The work presented in this thesis in terms of intention detection via gaze detection can be regarded as a first step towards this goal. As the hardware performance of the robots continues to improve, it is envisaged that these mobile assistive robots will become pervasive in our daily life. I feel privileged to be able to investigate some of the key issues related to the future applications of such robots and will continue to do so for my future career.

Bibliography

1. *Population ageing in Europe Facts, implications and policies*. 2014.
2. Pensions, D.f.W.a., *Family Resources Survey 2012/13*. 2014.
3. Gavrilov, L.A., *Demographic Determinants of Populations Aging*. DESA, 2003.
4. Surgical, I. *da Vinci Surgical System*. 2015; Available from:
<https://www.intuitivesurgical.com/company/>.
5. Surgical, I., *da Vinci Surgery*. 2014.
6. Health, I. *RP-7i Robot*. 2012; Available from: <http://www.intouchhealth.com/products-and-services/products/rp-7i-robot/>.
7. Butter, M.R., A. · Boxsel, J. van · Kalisingh, S. · Schoone, M. · Leis, M. · Gelderblom, G.J. · Cremers, G. · Wilt, M. de · Kortekaas, W. · Thielmaan, A. · Cuhls, K. · Sachinopoulou, A. · Korhonen, I., *Robotics for healthcare - Final report*. 2008, TNO Informatie- en Communicatietechnologie · TNO, The Netherlands Vilans, The netherlands EuroAct, Japan FhG ISI, Germany VTT, Finland.
8. Kononenko, I., *Inductive and Bayesian Learning in Medical Diagnosis*. Vol. 7. 1993. 317-337.
9. De Fauw, J., et al., *Automated analysis of retinal imaging using machine learning techniques for computer vision*. F1000Research, 2016. **5**(1573).
10. Shibata, T., K. Wada, and K. Tanie. *Subjective evaluation of a seal robot at the National Museum of Science and Technology in Stockholm*. in *The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003*. 2003.
11. Tapus, A. *Improving the Quality of Life of People with Dementia through the Use of Socially Assistive Robots*. in *2009 Advanced Technologies for Enhanced Quality of Life*. 2009.
12. Abdi, J., et al., *Scoping review on the use of socially assistive robot technology in elderly care*. BMJ Open, 2018. **8**(2).
13. Health, I.t. *JustoCat*. 2016; Available from:
<http://innovationtohealth.co.uk/justocat/?rq=JustoCat>.

14. Thrun, S., et al. *MINERVA - a second-generation museum tour-guide robot*. in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999.
15. Pollack, M.E., et al., *Pearl: A mobile robotic assistant for the elderly*. AAAI Workshop on Automation Caregiver: The Role of Intelligent Technology in ElderCare, 2002.
16. InTouchHealth. *RP-Vita Remote Presence Robot*. 2014; Available from: <http://www.intouchhealth.com/products-and-services/products/rp-vita-robot/>.
17. Spectrum, I. *How Aldebaran Robotics Built Its Friendly Humanoid Robot, Pepper*. 2014; Available from: <https://spectrum.ieee.org/robotics/home-robots/how-aldebaran-robotics-built-its-friendly-humanoid-robot-pepper>.
18. Tanaka, F., et al. *Pepper learns together with children: Development of an educational application*. in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. 2015.
19. Zhang, L., et al., *Powering the world's robots—10 years of ROS*. Vol. 2. 2017. eaar1868.
20. Thrun, S. and M. Montemerlo, *The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures*. Int. J. Rob. Res., 2006. **25**(5-6): p. 403-429.
21. Elfes, A., *Occupancy grids: a probabilistic framework for robot perception and navigation*. 1989, Carnegie Mellon University. p. 190.
22. Hahnel, D., et al. *An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements*. in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. 2003.
23. Stachniss, C., D. Hahnel, and W. Burgard. *Exploration with active loop-closing for FastSLAM*. in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. 2004.
24. Montemerlo, M., et al., *FastSLAM 2.0 - An improved particle filtering algorithm for simultaneous localisation and mapping that provably converges*, in *Proceedings of the 18th international joint conference on Artificial intelligence*. 2003, Morgan Kaufmann Publishers Inc.: Acapulco, Mexico. p. 1151-1156.

25. Grisetti, G., C. Stachniss, and W. Burgard. *Improving Grid-based SLAM with Rao-Blackwellised Particle Filters by Adaptive Proposals and Selective Resampling*. in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2005.
26. Grisetti, G., et al., *Fast and accurate SLAM with Rao-Blackwellised particle filters*. *Robotics and Autonomous Systems*, 2007. **55**(1): p. 30-38.
27. Moravec, H.P. and A. Elfes. *High resolution maps from wide angle sonar*. in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. 1985.
28. Yamauchi, B. *A frontier-based approach for autonomous exploration*. in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. 1997.
29. Thrun, S., *Learning Occupancy Grid Maps with Forward Sensor Models*. *Autonomous Robots*, 2003. **15**(2): p. 111-127.
30. Wan, T., et al. *Mobile robot simultaneous localisation and mapping in unstructured environments*. in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*. 2010.
31. Kohlbrecher, S., et al. *A flexible and scalable SLAM system with full 3D motion estimation*. in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. 2011.
32. Wang, C.-C., et al., *Simultaneous Localisation, Mapping and Moving Object Tracking*. *Int. J. Rob. Res.*, 2007. **26**(9): p. 889-916.
33. Microsoft, *Introducing Kinect for Xbox 360*. 2012.
34. LiDAR, V. 2015; Available from: <http://velodynelidar.com/>.
35. Davison, A.J., et al., *MonoSLAM: Real-Time Single Camera SLAM*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. **29**(6): p. 1052-1067.
36. Kaess, M. and F. Dellaert, *Visual SLAM with a Multi-Camera Rig*. College of Computing, Georgia Institute of Technology - Technical Report GIT-GVU-06-06, 2006.

37. Karlsson, N., et al. *The vSLAM Algorithm for Robust Localisation and Mapping*. in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. 2005.
38. Se, S., D. Lowe, and J. Little. *Vision-based mobile robot localisation and mapping using scale-invariant features*. in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. 2001.
39. Civera, J., A.J. Davison, and J.M.M. Montiel, *Inverse Depth Parametrisation for Monocular SLAM*. *IEEE Transactions on Robotics*, 2008. **24**(5): p. 932-945.
40. Payá, L., A. Gil, and Ó. Reinoso, *A State-of-the-Art Review on Mapping and Localisation of Mobile Robots Using Omnidirectional Vision Sensors*. Vol. 2017. 2017. 1-20.
41. Lienhart, R. and J. Maydt. *An extended set of Haar-like features for rapid object detection*. in *Image Processing. 2002. Proceedings. 2002 International Conference on*. 2002.
42. Dalal, N. and B. Triggs. *Histograms of oriented gradients for human detection*. in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. 2005.
43. Xavier, J., et al. *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.
44. Arras, K.O., O.M. Mozos, and W. Burgard. *Using Boosted Features for the Detection of People in 2D Range Data*. in *Robotics and Automation, 2007 IEEE International Conference on*. 2007.
45. Carballo, A., A. Ohya, and S. Yuta. *People detection using range and intensity data from multi-layered Laser Range Finders*. in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. 2010.
46. Shackleton, J., B. Vanvoorst, and J. Hesch. *Tracking People with a 360-Degree Lidar*. in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*. 2010.

47. Spinello, L., et al., *A Layered Approach to People Detection in 3D Range Data*, in *AAAI Conference on Artificial Intelligence 2010 (AAAI)*. 2010.
48. Bellotto, N. and H. Hu, *Multisensor-Based Human Detection and Tracking for Mobile Service Robots*. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009. **39**(1): p. 167-181.
49. Munaro, M., F. Basso, and E. Menegatti. *Tracking people within groups with RGB-D data*. in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. 2012.
50. Munaro, M. and E. Menegatti, *Fast RGB-D people tracking for service robots*. Autonomous Robots, 2014. **37**(3): p. 227-242.
51. Nash, A., et al., *Theta* - Any-angle path planning on grids*, in *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*. 2007, AAAI Press: Vancouver, British Columbia, Canada. p. 1177-1183.
52. Ferguson, D. and A. Stentz, *Field D*: An Interpolation-Based Path Planner and Replanner*, in *Robotics Research*, S. Thrun, R. Brooks, and H. Durrant-Whyte, Editors. 2007, Springer Berlin Heidelberg. p. 239-253.
53. Phillips, M. and M. Likhachev. *SIPP: Safe interval path planning for dynamic environments*. in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011.
54. Narayanan, V., M. Phillips, and M. Likhachev. *Anytime Safe Interval Path Planning for dynamic environments*. in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012.
55. Yuan, F., L. Twardon, and M. Hanheide. *Dynamic path planning adopting human navigation strategies for a domestic mobile robot*. in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. 2010.
56. Blackmore, L., M. Ono, and B.C. Williams, *Chance-Constrained Optimal Path Planning With Obstacles*. IEEE Transactions on Robotics, 2011. **27**(6): p. 1080-1094.

57. Fulgenzi, C., et al. *Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian processes*. in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008.
58. Luders, B.D., M. Kothariyand, and J.P. How, *Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty*. Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2010.
59. Henry, P., et al. *Learning to navigate through crowded environments*. in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2010.
60. Rasmussen, C.E. and C.K.I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. 2005: The MIT Press.
61. Khatib, O. *Real-time obstacle avoidance for manipulators and mobile robots*. in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. 1985.
62. Borenstein, J. and Y. Koren, *The vector field histogram-fast obstacle avoidance for mobile robots*. IEEE Transactions on Robotics and Automation, 1991. 7(3): p. 278-288.
63. Ulrich, I. and J. Borenstein. *VFH-star: local obstacle avoidance with look-ahead verification*. in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. 2000.
64. Fox, D., W. Burgard, and S. Trun, *The dynamic window approach to collision avoidance*. Robotics and Automation Magazine, IEEE, 1997. 4(1): p. 23-33.
65. Brock, O. and O. Khatib. *High-speed navigation using the global dynamic window approach*. in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999.
66. Ogren, P. and N.E. Leonard, *A convergent dynamic window approach to obstacle avoidance*. IEEE Transactions on Robotics, 2005. 21(2): p. 188-195.
67. Kiss, D. and G. Tevesz. *A receding horizon control approach to obstacle avoidance*. in *Applied Computational Intelligence and Informatics (SACI), 2011 6th IEEE International Symposium on*. 2011.

68. x00C, et al. *Investigation of Dynamic Window based navigation algorithms on a real robot.* in *Applied Machine Intelligence and Informatics (SAMI), 2013 IEEE 11th International Symposium on.* 2013.
69. Seder, M., K. Macek, and I. Petrovic. *An integrated approach to real-time mobile robot control in partially known indoor environments.* in *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.* 2005.
70. Seder, M. and I. Petrovic. *Dynamic window based approach to mobile robot motion control in the presence of moving obstacles.* in *Proceedings 2007 IEEE International Conference on Robotics and Automation.* 2007.
71. Javier, M. and L. Montano, *Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios.* IEEE Transactions on Robotics and Automation, 2004. **20**(1): p. 45-59.
72. Fiorini, P. and Z. Shiller, *Motion Planning in Dynamic Environments Using Velocity Obstacles.* The International Journal of Robotics Research, 1998. **17**(7): p. 760-772.
73. Gal, O., Z. Shiller, and E. Rimon. *Efficient and safe on-line motion planning in dynamic environments.* in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on.* 2009.
74. Quinlan, S. and O. Khatib. *Elastic bands: connecting path planning and control.* in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on.* 1993.
75. Fraichard, T. and H. Asama. *Inevitable collision states. A step towards safer robots?* in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on.* 2003.
76. Bouraine, S., T. Fraichard, and H. Salhi. *Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments.* in *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* 2012.
77. Althoff, D., et al. *Probabilistic collision state checker for crowded environments.* in *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* 2010.

78. Althoff, D., D. Wollherr, and M. Buss. *Safety assessment of trajectories for navigation in uncertain and dynamic environments*. in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011.
79. Toit, N.E.D. and J.W. Burdick, *Probabilistic Collision Checking With Chance Constraints*. IEEE Transactions on Robotics, 2011. **27**(4): p. 809-815.
80. Toit, N.E.D. and J.W. Burdick, *Robot Motion Planning in Dynamic, Uncertain Environments*. IEEE Transactions on Robotics, 2012. **28**(1): p. 101-115.
81. Tychonievich, L.A., R.P. Burton, and L.P. Tychonievich. *Versatile reactive navigation*. in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009.
82. Robots, A.M. *Pioneer 3-AT*. 2015; Available from:
<http://www.mobilerobots.com/ResearchRobots/P3AT.aspx>.
83. Komma, P., C. Weiss, and A. Zell. *Adaptive bayesian filtering for vibration-based terrain classification*. in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. 2009.
84. Weiss, C., et al., *Comparison of Different Approaches to Vibration-based Terrain Classification*. 2007.
85. Dupont, E.M., et al., *Frequency response method for terrain classification in autonomous ground vehicles*. Auton. Robots, 2008. **24**(4): p. 337-347.
86. Giguere, P. and G. Dudek, *A Simple Tactile Probe for Surface Identification by Mobile Robots*. Robotics, IEEE Transactions on, 2011. **27**(3): p. 534-544.
87. Weiss, C., H. Frohlich, and A. Zell. *Vibration-based Terrain Classification Using Support Vector Machines*. in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. 2006.
88. Coyle, E., E.G. Collins, Jr., and R.G. Roberts. *Speed independent terrain classification using Singular Value Decomposition Interpolation*. in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011.

89. Roy, N., G. Dudek, and P. Freedman. *Surface sensing and classification for efficient mobile robot navigation*. in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. 1996.
90. Hosoda, K., Y. Tada, and M. Asada, *Anthropomorphic robotic soft fingertip with randomly distributed receptors*. *Robotics and Autonomous Systems*, 2006. **54**(2): p. 104-109.
91. Mukaibo, Y., et al. *Development of a Texture Sensor Emulating the Tissue Structure and Perceptual Mechanism of Human Fingers*. in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2005.
92. Baudois, F.d.B.A.C.G.A.B.G.A.D.D.A.C.S.A.D., *Tactile texture recognition with a 3-axial force {MEMS} integrated artificial finger*. *Proceedings of Robotics: Science and Systems*, 2009.
93. Fishel, J.A. and G.E. Loeb, *Bayesian exploration for intelligent identification of textures*. *Frontiers in Neurorobotics*, 2012. **6**.
94. Yanco, H.A. and J. Drury. *Classifying human-robot interaction - An updated taxonomy*. in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. 2004.
95. Hillman, M. *Rehabilitation robotics from past to present - A historical perspective* in *The ICORR 2003(The Eighth International Conference on Rehabilitation Robotics), Proceedings of 2003*.
96. Scholtz, J.C., *Human-Robot Interactions: Creating synergistic cyber forces*, in *Multi-robot Systems: From Swarms to Intelligent Automata*, A.C. Schultz and L.E. Parker, Editors. 2002, Kluwer Academic Publishers. p. 177-184.
97. K.T. Ulrich, T.T.T., J. P. Donoghue, W.T. Townsend, *Intrinsically Safer Robots*, F.R.N.C. NAS10-12178, Editor. 1995: Barrett Technology Inc.
98. Bicchi, A., et al. *Physical human-robot interaction: Dependability, safety, and performance*. in *Advanced Motion Control, 2008. AMC '08. 10th IEEE International Workshop on*. 2008.

99. Yamada, Y., et al. *Construction of a human/robot coexistence system based on a model of human will-intention and desire*. in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999.
100. Bigge, B. and I.R. Harvey, *Programmable springs - Developing actuators with programmable compliance for autonomous robots*. *Robotics and Autonomous Systems*, 2007. **55**(9): p. 728-734.
101. Meka. *A2 Compliant Arm - A Dexterous Manipulator to Work With People*. 2012; Available from: <http://mekabot.com/products/compliant-arm/>.
102. Kaminaga, H., et al. *Backdrivable miniature hydrostatic transmission for actuation of anthropomorphic robot hands*. in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*. 2007.
103. Nickel, K. and R. Stiefelhagen, *Visual recognition of pointing gestures for human-robot interaction*. *Image and Vision Computing*, 2007. **25**(12): p. 1875-1884.
104. Axenbeck, T., et al. *Recognising complex, parameterised gestures from monocular image sequences*. in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*. 2008.
105. Viola, P. and M.J. Jones, *Robust Real-Time Face Detection*. *International Journal of Computer Vision*, 2004. **57**(2): p. 137-154.
106. George, A. and A. Routray, *A Fast and Accurate Algorithm for Eye Localisation for Gaze Tracking in Low Resolution Images*. Vol. 10. 2016.
107. Benfold, B. and I. Reid, *Guiding Visual Surveillance by Tracking Human Attention*. *Proceedings of the 20th British Machine Vision Conference*, 2009.
108. Benfold, B. and I.D. Reid. *Colour Invariant Head Pose Classification in Low Resolution Video*. in *BMVC*. 2008.
109. Sheikhi, S. and J.-M. Odobez, *Recognising the visual focus of attention for human robot interaction*, in *Proceedings of the Third international conference on Human Behavior Understanding*. 2012, Springer-Verlag: Vilamoura, Portugal. p. 99-112.

110. Stiefelhagen, R. *Tracking focus of attention in meetings*. in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*. 2002.
111. Hodgkinson, I.J., P.B. Greer, and A.C.B. Molteno, *Point-spread function for light scattered in the human ocular fundus*. Journal of the Optical Society of America, 1994. **11**(2): p. 479-486.
112. Haro, A., M. Flickner, and I. Essa. *Detecting and tracking eyes by using their physiological properties, dynamics, and appearance*. in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. 2000.
113. Gullstrand, A., *Helmholtz's Treatise on Physiological Optics*. The Optical Society of America, 1924.
114. Morimoto, C.H., A. Amir, and M. Flickner. *Detecting eye position and gaze from a single camera and 2 light sources*. in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. 2002.
115. Zhu, Z. and Q. Ji, *Eye and gaze tracking for interactive graphic display*. Mach. Vision Appl., 2004. **15**(3): p. 139-148.
116. Czy, A., et al. *Comparison of developed gaze point estimation methods*. in *New Trends in Audio and Video / Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2008*. 2008.
117. Lee, J.W., et al., *3D gaze tracking method using Purkinje images on eye optical model and pupil*. Optics and Lasers in Engineering, 2012. **50**(5): p. 736-751.
118. Chung, D.H.Y.a.M.J., *A novel non-intrusive eye gaze estimation using cross-ratio under large head motion*. Computer Vision and Image Understanding, Special Issue on Eye Detection and Tracking, 2005: p. 25-51.
119. Tomasello, M., et al., *Understanding and sharing intentions - The origins of cultural cognition*. Behavioral and Brain Sciences, 2005(28): p. 675-735.
120. Demiris, Y., *Prediction of intent in robotics and multi-agent systems*. Cognitive Processing, 2007. **8**(3): p. 151-158.
121. Kindermann, R. and J.L. Snell, *Markov Random Fields and Their Applications*. 1980: AMS.

122. Jain, A.K., Y. Zhong, and M.-P. Dubuisson-Jolly, *Deformable template models: A review*. Signal Processing, 1998. **71**(2): p. 109-129.
123. Roweis, S. and Z. Ghahramani, *A unifying review of linear Gaussian models*. Neural Comput., 1999. **11**(2): p. 305-345.
124. Buxton, H., *Learning and understanding dynamic scene activity - A Review*. Image and Vision Computing, 2003. **21**(1): p. 125-136.
125. Arkin, R.C., *Behavior-based robotics*. 1998: MIT press.
126. Acosta-Calderon, C.A. and H. Hu, *Robot Imitation - Body schema and body percept*. ABBI, 2005. **2**(3-4): p. 131-148.
127. Pezzulo, G. and G. Calvi, *A Schema Based Model of the Praying Mantis*, in *From Animals to Animats 9*, S. Nolfi, et al., Editors. 2006, Springer Berlin Heidelberg. p. 211-223.
128. Robots, A.M. *Pioneer P3-DX*. 2015; Available from:
<http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx>.
129. Health, D.O., *Health Building Note 00-04 -*
Circulation and communication
spaces. 2013.
130. Asfour, T., et al. *ARMAR-III - An Integrated Humanoid Platform for Sensory-Motor Control*. in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. 2006.
131. Johnson, A.E. and M. Hebert, *Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999. **21**(5): p. 433-449.
132. Guang-Bin, H., Z. Qin-Yu, and S. Chee-Kheong. *Extreme learning machine - A new learning scheme of feedforward neural networks*. in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. 2004.
133. Cambria, E., et al., *Extreme Learning Machines [Trends & Controversies]*. Intelligent Systems, IEEE, 2013. **28**(6): p. 30-59.

134. Chacko, B.P. and A.P. Babu. *Online sequential extreme learning machine based handwritten character recognition*. in *Students' Technology Symposium (TechSym), 2011 IEEE*. 2011.
135. Lucas, B.D. and T. Kanade, *An iterative image registration technique with an application to stereo vision*, in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. 1981, Morgan Kaufmann Publishers Inc.: Vancouver, BC, Canada. p. 674-679.
136. Weinrich, C., et al. *People detection and distinction of their walking aids in 2D laser range data based on generic distance-invariant features*. in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. 2014.
137. Schapire, R.E. and Y. Singer, *Improved Boosting Algorithms Using Confidence-rated Predictions*. *Machine Learning*. **37**(3): p. 297-336.
138. Pfaff, P., W. Burgard, and D. Fox, *Robust Monte-Carlo Localisation Using Adaptive Likelihood Models*, in *European Robotics Symposium 2006*, H. Christensen, Editor. 2006, Springer Berlin Heidelberg. p. 181-194.
139. Ba-Ngu, V. and M. Wing-Kin, *The Gaussian Mixture Probability Hypothesis Density Filter*. *Signal Processing, IEEE Transactions on*, 2006. **54**(11): p. 4091-4104.
140. Mahler, R.P.S., *Multitarget Bayes filtering via first-order multitarget moments*. *IEEE Transactions on Aerospace and Electronic Systems*, 2003. **39**(4): p. 1152-1178.
141. Likhachev, M., G.J. Gordon, and S. Thrun, *ARA*: Anytime A* with Provable Bounds on Sub-Optimality*, in *NIPS*. 2003.
142. Silver, D., *Cooperative pathfinding*, in *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2010, AAAI Press: Marina del Rey, California. p. 117-122.
143. Kushleyev, A. and M. Likhachev. *Time-bounded lattice for efficient planning in dynamic environments*. in *2009 IEEE International Conference on Robotics and Automation*. 2009.
144. McKeague, S.J., *Multi-Sensor Fusion for Human-Robot Interaction in Crowded Environments*. 2014, Imperial College of London.

145. Dodgson, N.A., *Variation and extrema of human interpupillary distance*. SPIE 5291, Stereoscopic Displays and Virtual Reality Systems XI, 2004.
146. Yang, G.-Z., et al., *The grand challenges of Science Robotics*. Science Robotics, 2018. **3**(14).

If you are going through Hell, keep going