IMPROVING LIBRARY MATERIAL SHELVING TIME BY IMPLEMENTING AN AUTONOMOUS BOOK TRUCK

presented by

JAN JACOBUS SPIES

Dissertation submitted in fulfilment of the requirements for the degree

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

in the

Department of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering, Built Environment and Information Technology

at the

Central University of Technology, Free State

Study leader: Dr B.J. Kotze

August 2020

Declaration

I, JAN JACOBUS SPIES, identity number , and student number , do hereby declare that this research project which has been submitted to the Central University of Technology, Free State, for the degree Master of Engineering in Electrical Engineering, is my own independent work and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State, and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.

18 August 2020

SIGNATURE OF STUDENT DATE

Acknowledgements

I would first like to thank my advisor Dr Ben Kotze of the Department of Electrical, Electronic and Computer Engineering at the Central University of Technology, Free State. The door to his office was always open whenever I had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to thank the following people and institutions. Without their participation, input and support, this research effort could not have been successfully conducted:

- Hannes Retief
- Clinton Agnew
- The library staff at the Soshanguve campus of TUT
- The Tshwane University of Technology

Finally, I must express my very profound gratitude to my Wife and son for providing me with unfailing support and continuous encouragement. This accomplishment would not have been possible without them. Thank you.

Author

Jan Spies

Abstract

The prompt shelving of returned library books is an important task in any traditional library. To help speed up the shelving process, this dissertation proposes an automated book truck capable of moving returned library books from the return desk back to the shelves.

By making use of the design and creation research methodology, software algorithms, sensors and robotic hardware are evaluated and then selected to construct an autonomous book truck. It is determined that an autonomous book truck should consist of a robotic body that has the same footprint as an average human. Furthermore, the sensor skirt should consist of at least a LIDAR or equivalent sensor to be used for obstacle avoidance and that sonar sensors should be used for localisation.

A simulator is created to test the selected components with the simulation data suggesting that shelving time – and therefore the dead time of returned books – is reduced by a significant factor. The research also provides a possible prototype which can be used for further development.

Table of Content

List of Figures

List of Tables

Glossary

A priori	The Latin phrase meaning “from the earlier” or “from before”, interpreted as already known knowledge, typically used when referring to a map of the environment which is known to the robot.
Actuator	A component of a machine that converts electrical energy into motion. In this study, the term refers to DC motors.
Autonomous Mobile Robot	Robots that can change their location through locomotion and which are not limited to previously set fixed routes.
Autonomous Guided Vehicle	Vehicles operating in specially modified environments, carrying out transportation tasks along fixed routes.
Behaviour Based Control	A control method making use of behaviours.
Belief	The most probable localised position of an entity in the real world.
Book truck	A trolley used to move books from the lending desk back to relevant shelves before the books are manually placed back into the shelves.
Crosstalk	A phenomenon encountered when using multiple ultrasonic sensors firing close to each other. Signals from one ultrasonic sensor are picked up by other sensors and then interpreted as being sent from the second sensor.
Dead time	The time that it takes from when a book is logged into the LIS after being returned, to when it is shelved and available for re-issue.

Exteroceptive Sensors	Sensors that are on-board a robot but are gathering data from the world the robot moves in, i.e. range finders and cameras.
Global Navigation	Initial path planning implemented on a higher level which does not contain full environmental information. Initial navigation is done on a room-to-room level which can subsequently be refined to avoid and navigate obstacles in the robot's immediate environment.
Infrared	Referring to infrared light used in obstacle avoidance sensors.
JPG / JPEG	It is a file extension for a lossy graphics file and was created by the Joint Photographic Experts Group.
Library Information System	The electronic cataloguing system used by a library to manage books and other resources, users, and to gather data and statistics regarding lending patterns, etc.
Library User	see Patron
Patron	A person visiting the library, making use of library facilities.
Proprioceptive Sensors	Sensors on-board the robot gathering local data, i.e. inertial units and odometry.
Robotic Work Cell	An area wherein a manipulator robot works.
Sensor Skirt	A ring of sensors covering the robot.
Service Robot	An electro-mechanical machine, guided by a computer program and/or electronic circuitry, assisting humans in day-to-day tasks.

Shelver	Library staff members, permanent or temporary, responsible for placing books in the correct places on the library shelves.
SSoc	The open collection of books at the Soshanguve South campus of the Tshwane University of Technology.
User	see Patron
World	The area or map that the robot will be working or moving in.

Acronyms

AGV	Autonomous Guided Vehicle
AMR	Autonomous Mobile Robot
AL	Algorithm Layer
AuRoSS	Autonomous Robotic Shelf Scanning Systems
CAD	Computer Aided Design
DL	Driver Layer
FOV	Field of View
ILS	Integrated Library System
IR	Infra-Red Light
JPG / JPEG	Joint Photographic Experts Group
LIS	Library Information System
LUCAS	Limerick University Computerized Assistive Systems
MCL	Monte Carlo Localization
NI	National Instruments
OCR	Optical Character Recognition
PL	Platform Layer
ROS	Robotic Operating System
SLAM	Simultaneous Localization and Mapping
SSoc	Soshanguve South open collection

TUT	Tshwane University of Technology
UIL	User Interface Layer
US	Ultrasonic Sensors

Chapter 1 – Introduction

This chapter provides a background to the research and justification of the research topic. It also provides the research goals, objectives and the research methods used throughout this dissertation.

1.1 Background

Libraries are an integral part of our society's knowledge repository. Even though technological advances such as the internet, smart devices and an 'always-connected-society' provide avenues for fast and almost instantaneous access to knowledge, libraries still provide a physical place for the collection and dissemination of knowledge.

One of the most frequently listed complaints among library users is that the electronic cataloguing system shows that the required book is available, but the book is not found on the shelf [1].

This complaint is due to the two-part process still being used in traditional libraries:

1) The user goes to the lending desk and presents the book which must be returned. The librarian enters the book details into the computerised lending system which updates the lending detail and shows the book as being available to be borrowed again, and places the book on a book truck. This part of the process is fast and efficient.

2) All the returned books at the lending desk must be physically moved to the shelves and placed in their respective places on the shelves. This is a manual process which is performed multiple times during each working day, when the book truck is full, or at the end of the working day. This part of the process is slow and inefficient.

Since checking the book back into the library inventory and the physical act of shelving the book does not happen consecutively, this leads to books showing as available in the library database, whilst not being physically available on the shelves.

A benchmarking report [2] submitted by a team at the University of Virginia found that library books take between 1 and 5 days to travel from the return desk to their relevant shelves. This report also shows that book pick-ups are done more frequently from the most obvious places, such as shelf-ends, and less frequently from the less obvious places, such as study cubicles. These book pick-up patterns contribute to the delays in relation to books being returned to their respective shelf positions. Focus group participants noted that this delay experienced between recording a book on the electronic catalogue and the time it takes before a book is shelved and therefore available for re-circulation increases user frustration [3].

Another study done on shelving concluded that shelving is a repetitive and physically demanding task, especially when trolleys full of books need to be moved from the circulation desk to the relevant shelves, which can lead to repetitive strain and overuse injuries [4][5].

1.2 Research Topic Justification

This research aims to address the issue of material dead time by suggesting the development and implementation of an autonomous mobile robot (AMR), as a mix between a human-surrogate and human-centred robot. This will 1) enable books to be moved quicker between the return desk and predefined drop-off points close to the book's relevant shelf and 2) free librarians and shelvers from pushing heavy book trolleys.

A service robot is described as a mechanical or virtual agent guided by a computer program or electronic circuit [6], while [5] and [7] define a service robot as a fully or semi-autonomous robot performing services useful to humans. However, this robot is not used in the manufacturing process.

Service robots are further divided into the following themes [7]:

- Human surrogate robotics: Where the robot substitutes the human in hostile and remote environments.
- Human-centred robotics: Where a robot works together with a human, in human environments.

- Human augmentation: Where the robot is essentially part of a human for example exoskeleton assistance and surgery.

These physically demanding tasks can be performed by one or more service robots, which will not only reduce shelving time but will also reduce staff injuries related to pushing heavy book trolleys.

1.3 Goals and Objectives

The research goals and the research objectives of this research effort consisted of the following:

1.3.1 Research Objectives

The focus of this research was to compare the circulation data, as retrieved from the Library Information System (LIS), with a simulation of the library environment and data gathered from an AMR in a real-world library environment.

The data gathered were used to draw conclusions on whether an AMR will reduce library material dead time.

The research was designed to:

a) Investigate current techniques used by mobile robots to navigate and move, and then identify the most appropriate techniques for a library environment.

b) Create a simulated environment of an operational library. This simulation will include a model of the robot incorporating the identified techniques to simulate the validity of the selected techniques.

c) Apply the selected techniques on a robotic platform to verify the simulated results

d) Draw conclusions on the effectiveness of an AMR in the library environment.

1.3.2 Research Questions

The primary research hypothesis in this study is:

> The implementation of an autonomous book truck, moving returned library material from the lending desk back to the shelves, will reduce library material dead time in a traditional library.

In an attempt to answer the hypothesis, the following research questions must be addressed:

- What will the robotic platform look like?
- Which sensors will be used to facilitate localisation and obstacle avoidance?
- Which mapping and localisation techniques or combination of techniques are the best to implement?
- Can the robot be implemented in a typical library setup without library remodelling?

1.3.3 Benefits of the Study

A problem area found in libraries is the time it takes for books to be returned to their proper shelf locations. These books are essentially unavailable although the electronic catalogue shows them as available.

Implementing an autonomous mobile robot would contribute to minimising library material dead time and would also minimise injuries relating to the repetitive nature of pushing heavy book trolleys.

1.3.4 Scope and Delimitations

The scope of the study is as follows:

- A literature study to review available library service robots.
- A literature study regarding service robot localisation techniques and selecting an appropriate localisation technique for this context.

- Building a model and simulation system to test the selected localisation technique and sensors in a software simulation environment.
- Implementing the simulated model on a mobile robot platform.
- Testing and evaluation will take place in a simulated and/or real library environment. The success of the robot will be determined by its successful navigation of the shelves and the correct delivery of the books within specified time limits.
- To draw conclusions and make recommendations for future implementations of mobile robots in library environments.

The research was done with the following delimitations:

- This study will focus on the localisation and safe movement of the robot in a semi-structured environment.
- No mapping will be done, and an a priori map of the environment must be available.
- The robot will only deliver books to predetermined areas next to the shelves and will not attempt to shelve any of the books.
- The physical size/construction of the robot chassis might limit the number of books that can be moved at any one specific time.
- The mobile platform will not be able to access different floors in multi-level libraries.
- No battery charging station will be provided or simulated.
- Book identification will not be implemented.
- Data gathering using the AMR will be done when the library is closed to avoid possible collision with humans.

1.4 Method

The focus of this paper is to determine if an Autonomous Mobile Robot (AMR) can reduce the time it takes for library material to be shelved, in other words, reducing library material dead time. In support of this, book return data retrieved from the LIS are used to create time frames which are compared to simulation data and real-world data.

An AMR simulator is created that implements localisation, path finding and obstacle avoidance techniques. The simulation time is compared against the data obtained from the LIS.

The simulated model is also applied to an actual mobile robot to compare the simulated and real-world data.

The outcome of this study is a determination, based on a comparison between simulated and real-world data, of whether an AMR would decrease library material dead time.

1.5 Chapters Outline

The dissertation is divided into the following chapters:

Chapter 1: Introduction

> Provides an analysis of the background of the research and discusses the motivation behind the research, the research questions and the objectives.

Chapter 2: Literature Review and Methodology

> Presents a comprehensive description of the current robotic automation found in libraries. It also discusses current methods used for mapping, localisation and movement. The research methodology and the research design adopted in this research are also discussed.

Chapter 3: Methodology

This chapter explains the design of the simulator and the control theory used.

Chapter 4: Algorithm Layer and Hardware Design

This chapter focuses on the design criteria of an automated book truck platform.

Chapter 5: Results, Conclusions and Recommendations

Presents the results based on the research questions and draws conclusions on the research effort and offers recommendations based on the research effort.

1.6 Chapter Summary

In this chapter, the motivation for the research is discussed. It is proposed that an autonomous book truck might improve the turnaround time of library material, specifically books in the general section. The research design used to conduct the research is also discussed.

In the next chapter, the literature review is conducted with an in-depth view of robots and autonomous systems currently found in library environments. Current methods used to localise, find paths and avoid obstacles are discussed together with the research methodology used to conduct this study.

Chapter 2 – Robotic Automation Technology in Libraries

In this chapter the implemented and proposed automation systems specific to the use of robotic systems in libraries is discusses.

It further explores current mobile robot localisation, path planning and movement techniques and concludes with a discussion regarding best practices for robot control software and firmware.

2.1 Background

Libraries are not new to technological advances. The Libraries and Technology report from 1967 set out to analyse trends in the use of technology and to predict the future use of technology in libraries. This report states that: “The equipment with the greatest potential impact for library operations is computers”, specifically related to the cataloguing process and automated indexing and classification [8]. Computers have indeed shown their great potential in library services, specifically with computerised indexing and lending systems.

Even today, a lot of focus is still being placed on automated indexing and inventory control using RFID, as well as self-service check-in and check-out kiosks in order to streamline the user experience [9] [10].

Regardless of how books and inventory are managed, be it a manual, computer or RFID inventory and cataloguing systems, one aspect still has not changed: The traditional method used to deal with returned books. This traditional method still follows the same process: The library staff member receives a returned book and puts it on a book trolley, this book trolley gradually fills with books during the day. When it is full, the book trolley is pushed to the relevant shelves and the books are manually shelved by library staff. This traditional method leads to the problem being investigated – that of a library book showing as available on the electronic catalogue, but which might still be waiting to be shelved on a book trolley back at the lending desk.

2.2 Library Service Robots and Library Automation

Libraries are not new to technological advances and since libraries are the custodians of knowledge, it stands to reason that these institutions would be at the forefront of new technological adaptations.

This section discusses robotic automation in libraries and starts with the first robot used in a library. It continues by grouping the remaining research into two categories, 1) Robots used in off-site storage facilities, and 2) Robots used in libraries.

2.2.1 Harry

Arguably, the first robot that was introduced into a library environment was in 1994, in a Swedish library [11].

Harry (Figure 2-1), as the IRB2000 manipulator robot is known, is a six-axis immobile robotic arm within a highly structured environment. Library users would place returned books on a conveyer belt, optical scanners would scan bar codes attached to each book where after Harry executes pre-programmed movements to pick up the book, using specially designed grippers. After scanning and identifying the book, it was then placed into one of 18 pre-designated bins according to the predefined classification system (Figure 2-2). When these bins were full, they were removed and the books were then shelved by hand.

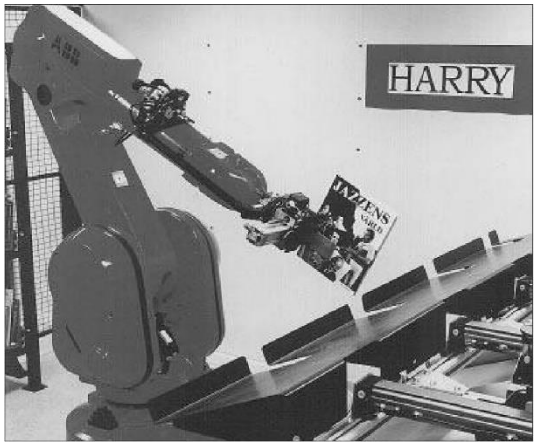


Figure 2-1: Harry at the conveyer belt

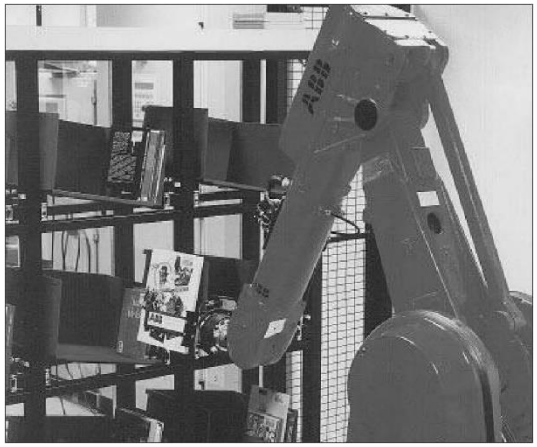


Figure 2-2: Harry at the designated bins

Harry was not able to move and therefore had no need for any mapping or localisation methods. All movements were pre-programmed depending on the identified book. Since it was in a controlled work cell, Harry also did not have the need for obstacle avoidance sensors or algorithms. Harry ultimately identified books by scanning attached bar codes and placing these identified books into designated containers until these containers were removed and the books shelved by library staff. Harry can sort the books according to specific criteria, but it is still up to library staff to shelve the books.

Not only did this robot sort books faster than the library staff, but also safeguarded them against repetitive strain injuries due to manually sorting the books.

2.2.2 Off-Site Storage

Since the growing number of printed materials has led to severe space constraints for libraries, some libraries decided to store selected material off-campus. These off-campus facilities might not be within walking distance from the main libraries or might not be suitable for a large number of library users at the same time. Although delivery options from remote locations do exist, the ability to browse these materials is reduced [12].

Since users should be able to view and browse this material from remote locations, several systems for remote library material viewing and browsing were introduced.

A book browsing system using a tele-operated autonomous robot to allow library users to select and browse through books in a remote location is introduced [13]. Library users do not need to know where the book is physically located since the robot has an a priori map of the library and the location where the book is shelved. After the library user supplies the title of the book, the robot can autonomously move to the shelf where the book is located at a speed of 300 mm/s. Using video feedback, a part of the shelf containing the relevant book is displayed and the library user would then select the wanted book using a tele-operation interface. YAMABICO (Figure 2-3) accomplishes book retrieval and remote browsing by using a purpose-built actuator with video feedback sending images of the open pages, back to the library user. The library user can then select to view new pages or return the book and select another book to browse.

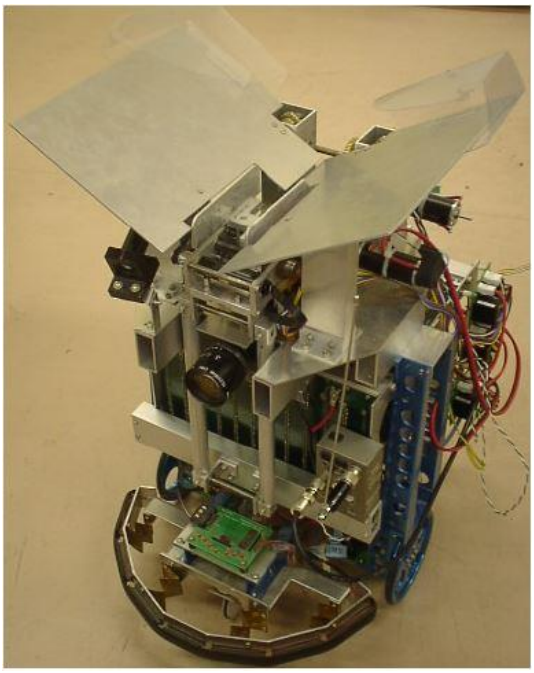


Figure 2-3: YAMABICO

Suthakorn et al. went a step further by proposing that the tele-operation-book-selection-step must be replaced with the robot retrieving the book from the shelf without any user input, using only book localisation techniques which might include image recognition techniques or RFID technology [12].

A work in progress report [15] introduced an autonomous librarian capable of finding a user specified book, retrieving the book and returning the found book to the user, whereas an intelligent book retrieval and return system based on a team of isolated robotic systems moving between dedicated shelves and identifying books using RFID technology was proposed [16].

2.2.3 In-Library Robots

The Limerick University Computerized Assistive Systems or “LUCAS” is an autonomous robotic aid, assisting library users to find books.

LUCAS (Figure 2-4) helps people to find the books they are looking for by escorting users directly to the shelf location of the specified book that is being searched for. LUCAS is a differential-drive robot, making use of an a priori map. Localisation is achieved by fusing

odometry, sonar and vision data and then comparing this data with map features. LUCAS does not physically move or manipulate books, but rather escorts a user to where a specific book is located [17].

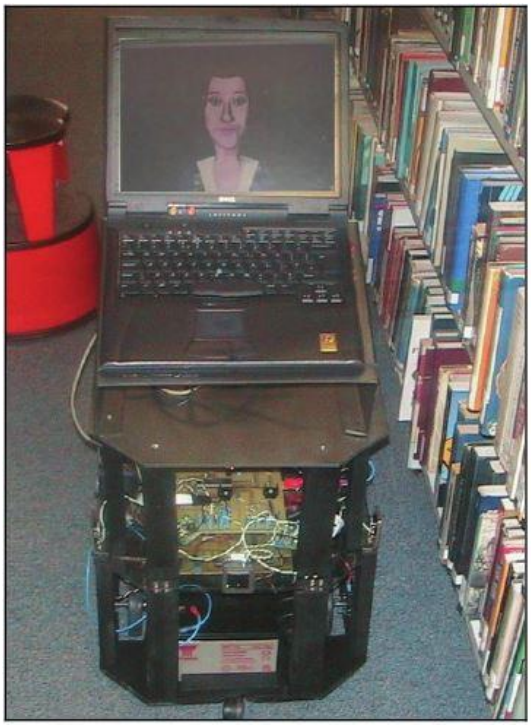


Figure 2-4: LUCAS

The UJI librarian robot (Figure 2-5) is used in the library at Universitat Jaume I (UJI) [18]. When users request a book, the UJI robot locates the book in the library using a vision system and a priori map of the library. The vision system is an eye-in-hand system which puts the stereo camera in line with the mechanical grippers. UJI uses the camera to identify the book and the alignment of the optical axis with the gripper axis facilitates easier book manipulation.

Figure 2-5: The UJI Librarian Robot

The UJI robot is a differential drive robot with a sensor skirt containing sonar sensors, infrared sensors, bumper sensors and a SICK laser rangefinder. These sensors are used for collision detection and navigation. A Mitsubishi manipulator is mounted on top of the platform on which the gripper and vision system are fitted. The manipulator arm is used to retrieve books from the shelf as requested by the user.

A study on what a child-friendly library robot might look like was conducted after which a prototype robot was designed to assist children with finding library resources [19]. Book Smile (Figure 2-6), as the robot is known, can recommend a book, locate a book and identify its position on the screen and also obtain/verify the book by instructing the patron to put the book into its “mouth”, allowing Book Smile to scan the book and verify that the child patron selected the correct book. The user could then continue looking for books or can ask the robot to return to the lending desk in order to check out the selected book/s. Book Smile is an omnidirectional robot, using human-sensitive sensors to not only avoid collisions but also to stay close to the user it is currently assisting.

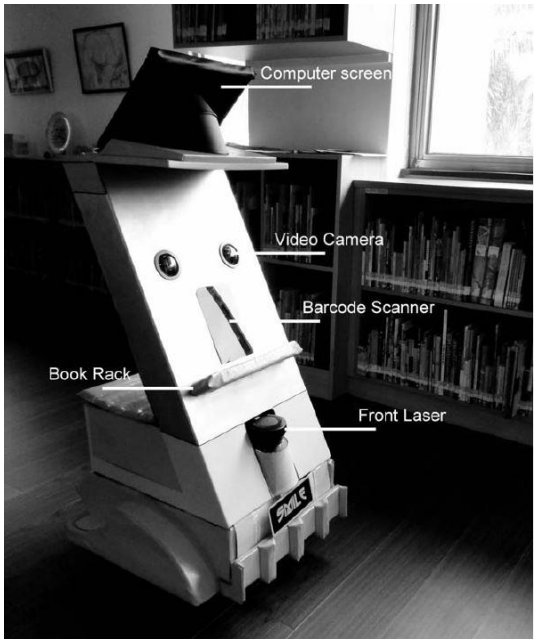


Figure 2-6: Book Smile

The New York Public Library installed a train system carrying user selected books between patrons and the storage area (Figure 2-7) [20][21]. This book train is not an autonomous system since library staff still needs to find and place selected books on the book train. The book train essentially just links two library areas together and provides a simplification to the library material retrieving and shelving process.

Figure 2-7: New York Library Book Train

Another large area of research is in techniques and methods used to locate and sort books using RFID technology, either as part of an autonomous system or in conjunction with mobile technology. These robots are capable of detecting books shelved incorrectly and also to verify available books [6][22].

Li et al. created AuRoSS (Autonomous Robotic Shelf Scanning Systems)(Figure 2-8)(Figure 2-9) which can scan the books in the shelves and is then able to determine if the books are in the correct sequence by verifying their position against the library's database. AuRoSS is a differential drive robot making use of an a priori map of the library to localise itself. It moves autonomously, at night, and produces a report on the missing and misplaced books. AuRoSS makes use of the bookshelf surfaces to plan its routes. Ultrasonic sensors attached to the arm, used to scan the books, ensure that the RF scanner is close enough to the shelf to successfully scan the books.

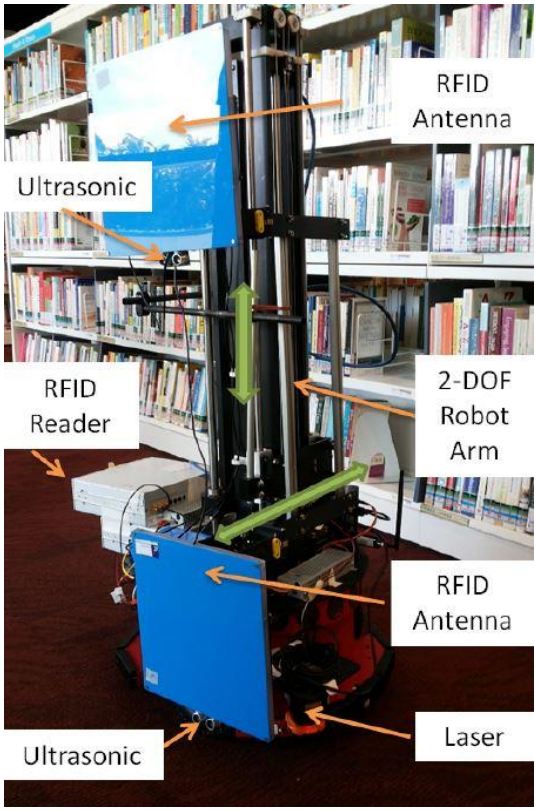


Figure 2-8: AuRoSS

Figure 2-9: AuRoSS

This section shows that although mobile robotics and related research is taking place in library environments, none of the research focuses on moving books from the lending desk BACK to the shelves.

2.3 Autonomous Robots and Robot Navigation

Localisation and navigation is one of the most important tasks for mobile robots [25]. Sariff and Buniyamin identifies three problems associated with navigation, namely: Localisation, path planning and motion control [26], while Borenstein simplifies the problem of navigation into the following three questions: “Where am I?”, “Where am I going?” and “How should I get there?” [27].

Nehmzow defines a mobile robot as a class of robots consisting of either Automated Guided Vehicles (AGV) or Autonomous Mobile Robots (AMR) [14]. AGVs operate in specially modified environments carrying out transportation tasks along fixed routes using beacons, induction loops or other markers. AMRs, however, can change their location using some means of locomotion. Their location is not limited to previously defined routes.

As stated, navigation is an important component of mobile robots. Without navigation, an AMR will not be able to reach its goal. This section will look at the current techniques used to answer the three questions of navigation as framed by Borenstein, which are: “Where Am I?”, “Where Am I Going?” and “How Should I Get There?” [27].

2.3.1 Mapping and Localisation Techniques

Mapping denotes the techniques used to represent the real world on a 2-dimentional flat surface. Apart from being able to represent the real world, the AMR should be able to place itself in this mapped world. Executing these actions with success allows the AMR to complete its goal.

This section will discuss mapping techniques and methods of localising the AMR in the real world and then to plot its position onto such a created map.

2.3.1.1 Creating a Map of the Environment

AMRs are either given the map of the world they are going to work in, called an a priori map, or they need to discover the map by themselves as they navigate, which is known as SLAM (Simultaneous Localisation and Mapping).

Two basic methods of representing a 2D environment are the configuration space method (Figure 2-10), where the dimensions of the environment and all the coordinates of the obstacles are known, and the occupancy grid method (Figure 2-11), where the environment is specified at a certain resolution and each block/tile represents either free space (white block) or occupied space (black block) [25] .

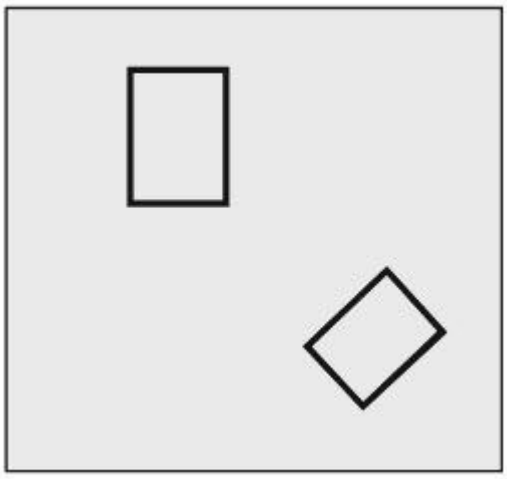


Figure 2-10: Configuration Space

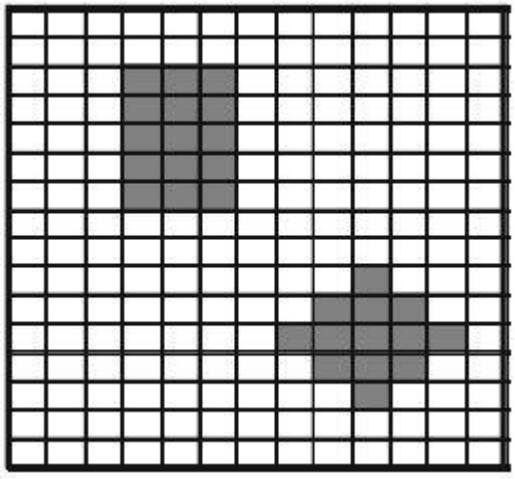


Figure 2-11: Occupancy Grid

An occupancy grid is a very common method of dividing an area into manageable tiles or blocks especially in the presence of noisy and uncertain sensor measurement data. Each tile is a piece of the map containing specific information regarding that area of the map.

2.3.1.2 Localisation

For an AMR to reach its goal or complete its tasks, it needs to have knowledge of its location, also called its *pose*. It needs to know where it is in the world that it occupies.

One of the earliest methods of localisation is the concept of dead reckoning, also known as odometry. This technique makes use of wheel movement in order to determine, through simple mathematical procedures, the vehicle's new position based on the previous position [25][28].

To determine wheel rotation, a wheel encoder is employed. This encoder is normally placed on the motor shaft to make use of the step-up effect of the gear ratio of the motor's gearbox. Although several different encoder sensors exist, the most common types are optical and magnetic decoders. Regardless of the detection method, the concept of operation stays the same.

Wheel encoders work by detecting transitions on a disc attached to the shaft of the motor. As the shaft turns, the disc turns in front of the sensor. Counting the transition pulses in a specific time frame is used to not only determine distance moved but also wheel velocity. The distance travelled is used to update the position of the vehicle while the wheel velocity is used to control the vehicle's movement speed.

Although dead reckoning is seen as the "backbone" of the majority of land-based mobile robots [28], it should be limited to short time usage since all sensor errors add up over time. Wheel slippage is especially bad since it has the largest effect on position accuracy [25].

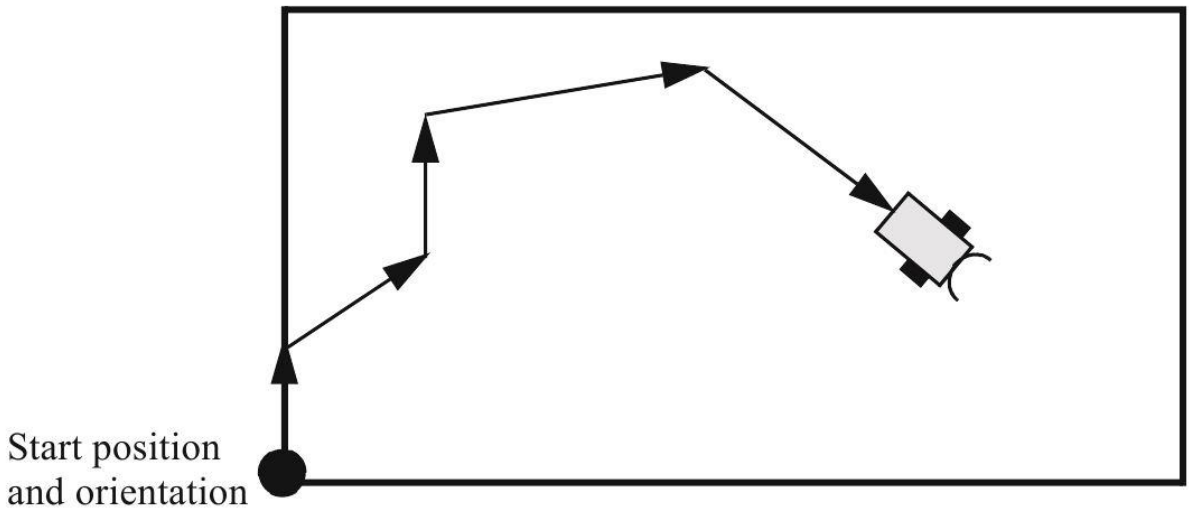


Figure 2-12: Dead reckoning

Figure 2-12 shows how dead reckoning achieves localisation. The movement of the vehicle is broken down into short segments which are then added together. The final location is therefore determined by adding all the movement vectors to the initial starting vector.

Burguera, González and Oliver states that probabilistic methods used to localise robots are the most promising when dealing with sensor data uncertainty and pose uncertainties in real-time [29].

According to Gutmann and Fox, probabilistic localisation is the process of determining the probability of the pose of a robot after taking sensor inputs into account and performing certain actions, for example, moving the robot. The concept of *belief* is used in probabilistic localisation to reflect the AMR's internal knowledge of its pose [30].

Localising using the Bayes group of filters is the most general approach to compute belief; however, the Bayes filter itself is not a tractable implementation for continuous state spaces. Several tractable implementations of Bayes filters exist of which Gaussian filters, particularly the Kalman filter, and Particle filters are the most common [29][31].

Kalman filters are sets of mathematical equations that support the estimation of the past, present and future state of a system or process [32] and although it has been successfully applied to AMR localisation, only one pose hypothesis can be represented, leading to the inability to globally localise or to recover from localisation failures [30][33].

Particle filter localisation achieves localisation by defining a number of ‘particles’ or virtual mobile platforms with the same characteristics as the actual mobile platform. These characteristics include error models of the sensors and error models of the kinematics of the robot platform. Particle filters solve the global localisation problem and are easy to implement and do not require a fixed computational time which alleviates hard real-time constraints [31].

The localisation process is a continuous execution of a scan-move cycle. Initially, particles with a random pose are distributed into the environmental model, except in locations where there are obvious obstacles, like walls, etc.

As part of the scan cycle, sensor data from actual robot are compared to virtual sensor data from the particles, taking the error models into account. This comparison determines the fitness of each particle by using the correlation between the sensor data from the actual robot and the virtual sensor data from each particle. The higher the correlation between the two sets of data, the bigger the probability that that specific particle represents the robot’s position in the world and the bigger the particle’s fitness. After calculating each particle’s probability function, re-sampling takes place to ensure that only particles with the highest fitness remain while removing those with very low probabilities. Robot pose is now estimated by using the probability density function of all the remaining particles.

During the move cycle, the movement vector applied to the actual robot is also applied to each of the particles. This ensures that particles execute the same movement as the actual robot and allow particles with large fitness values to ‘follow’ each other and create a probability density function.

Localisation and pose determination are achieved by either 1) averaging the probability density functions (PDF) of all the particles, or 2) by adopting the pose of the particle with the biggest belief [34][35].

This section describes the various mapping techniques and explains why the Occupancy Grid method is being favoured as method when using probabilistic localisation. It further shows that Particle Filters are superior to standard Kalman filters when localising an AMR.

Although an Occupancy Grid map represents the world as discrete areas, the AMR sees the world as a continuous space and is not limited to the middle of the discrete blocks. It can be anywhere in this continuous space.

The next section will look at how an AMR determines the path towards the goal, given a map, its location and the goal location.

2.3.2 Path Planning

Given a map and the goal location, path planning involves finding the shortest most efficient route from the current position of the AMR to the selected goal.

This Global method makes assumptions about the environment that are not always true, with the biggest assumption being that the environment is structured and constant. Modern robots usually operate in dynamic environments with absent or partial world data. This forces the global path planning to be associated with local obstacle detection and avoidance [36].

This section presents path planning techniques using the Global and Local concepts of path planning.

2.3.2.1 Global Path Planning

The starting point of the desired path is always the current pose of the robot, with the goal being the point that needs to be reached to complete its current task.

Goal selection for a robotic task depends on the task at hand. If the robot is moving equipment or parts around, the goal depends on where the parts must go. In the case of this study, the goal of the robot is dependent on the book loaded onto the robot.

According to Russell and Norvig, A-star search is the most widely known search algorithm. It will find a path if one exists and is also able to find the most optimal path. It is therefore both complete and optimal [37].

A-star evaluates nodes by combining the cost to reach the current node and the predicted cost from the current node to the goal. To determine these costs, a distance graph containing likely nodes must be created from the map of the world.

A distance graph describes the environment at a higher level and does not contain all the environmental information. Distance graphs contain only a few nodes and the distances relative to each other. Distance graphs are used to do the initial higher-level path planning where the robot moves between areas or rooms. Refined path planning can then be done after the robot reaches the desired area or room using various beacon detection techniques.

Distance graphs are created by implementing two distinct steps: 1) Identifying the separate nodes, and 2) Linking all the relevant nodes together.

Traditionally, each block in the occupancy grid map is identified as a node. However, this can lead to the following problems: 1) The number of nodes is very large and working with all these nodes might become infeasible in large environments; and 2) Path planning in an environment where each block is a node leads to suboptimal paths since only multiples of 45° or 90° turning angles are supported. Two of several methods that can be employed to identify valid node positions are 1) Visibility graphs and 2) Quad Trees.

Visibility graphs are normally, but not exclusively, used when the world is represented as a configuration space. Visibility graphs link the corners of obstacles together with an optional safety margin which can be half of the robot diameter. When the Start and Goal positions are included, a distance graph can be created that shows nodes linked from the Start to the Goal position.

Quad tree node generation takes the given environment and recursively divides the environment into four quadrants. If a quadrant is empty, a node is placed in the centre of the quadrant. If a quadrant is not empty, the non-empty quadrant is again divided into quadrants. This process repeats itself until all the quadrants are empty or until a specific resolution is reached [25].Figure 2-13 shows an Occupancy Grid Map of a fictitious environment before any node creation algorithms have been applied. The difference in the number of nodes created using the traditional node creation method (Figure 2-14) and creating nodes using the Visibility Graph (Figure 2-15) and Quad Tree (Figure 2-16) methods can be seen when comparing the number of nodes between the different methods (Table 2-1).

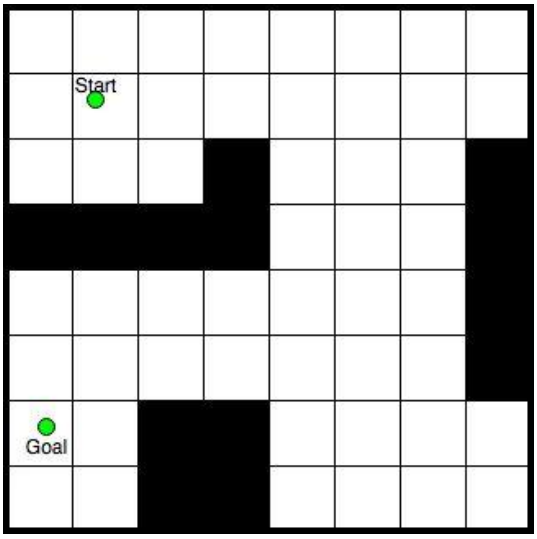


Figure 2-13: Occupancy Grid Map

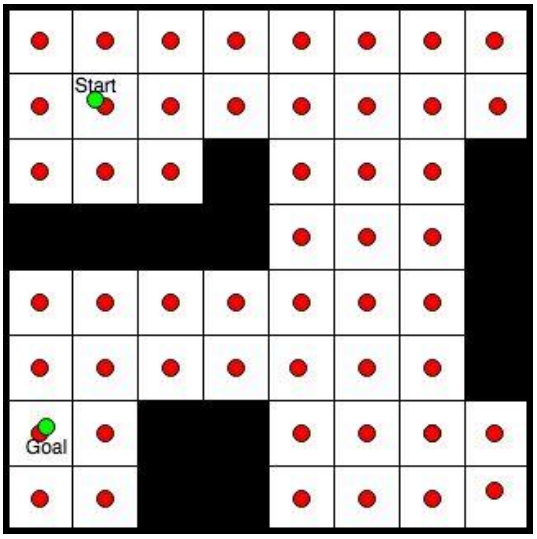


Figure 2-14: Nodes created using the Traditional Method

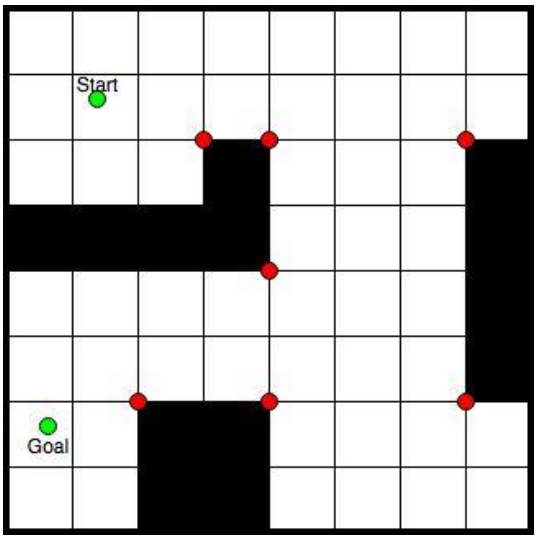


Figure 2-15: Nodes created using a Visibility Graph

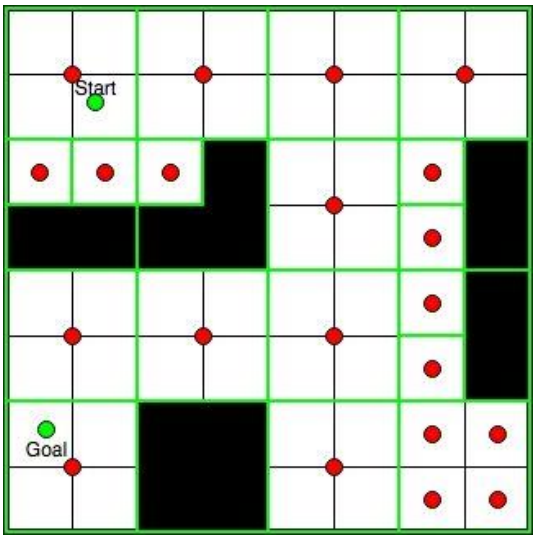


Figure 2-16: Nodes created using the Quad Tree Method

Table 2-1: Number of Nodes comparison table

Node Creation Algorithm	Number of Nodes including Start and Stop
Traditional	53
Visibility Graph	9
Quad Tree	23

After identifying the nodes, the final step in creating the Distance Graph is to connect all the nodes together and then to eliminate links between nodes if those links cross an obstacle or boundary. In other words, the link between two nodes is removed when there is no direct line because of a blocking obstacle or boundary.

The generated distance-graph, regardless of the method used, is now used by the path-finding algorithm to determine the best route to follow to reach the goal. The path planner creates a node list, containing a sequence of the nodes that need to be followed to reach the goal (Figure 2-17)[25].

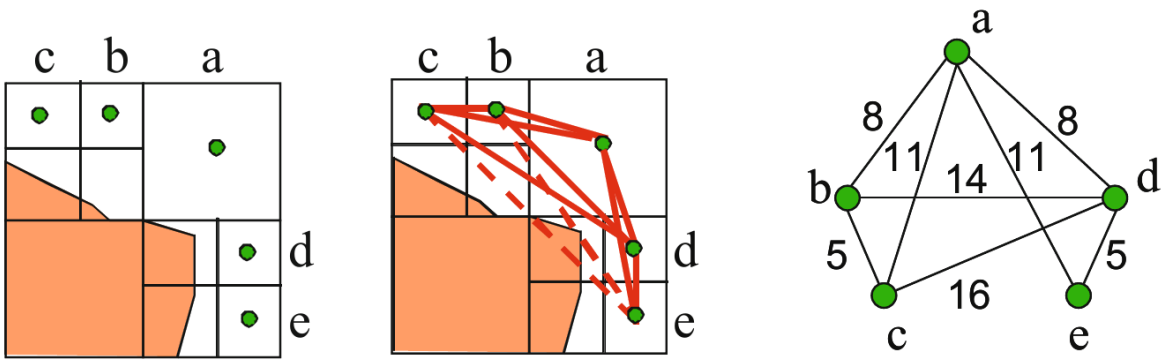


Figure 2-17: Node Tree Construction from Quadtree

Assuming no unknown obstacles are introduced into the environment, the global path will lead the robot to the goal. Each of the nodes will become an interim goal and when reached, the next node from the node list will be set as the goal, until the final node is reached. When the final node is reached, the robot will have reached its goal position.

2.3.2.2 Local Path Planning - Obstacle Avoidance

According to [36] and [38], obstacle avoidance is described as methodologies of shaping the robot's path, using information from its sensors to overcome unexpected obstacles.

The direction that the AMR must travel in is determined by combining the direction to the goal with the direction away from the detected obstacles. The ratio used to combine these vectors can be dynamically changed based on the distance to the detected goals. If the AMR is far away from obstacles, the combined path favours the direction of the goal and if the AMR

is close to the obstacles, the direction of travel favours the vector pointing away from all the obstacles.

Together with global path planning, obstacle avoidance is used to move a robot closer to its goal location by managing and navigating unknown obstacles in its immediate vicinity.

Some of the simplest obstacle avoidance techniques used in mobile robots are: 1) The Wandering Standpoint Algorithm, and 2) The Bug family of Algorithms.

The Wandering Standpoint Algorithm only requires a distance sensor and some sense of where the goal is. This algorithm tries to reach the goal by moving the robot in a straight line towards the goal. On detecting an obstacle, a decision is made to either turn left or right, based on the smallest avoidance angle, from where the robot then implements boundary following until the goal-direction is clear again. This process continues until the goal is reached [25].

The Bug family of algorithms consists of the Bug1, Bug2, DistBug or the Tangent Bug variations [25][38]. To use either of the Bug family algorithms, the robot's own position is needed together with either touch sensors or distance sensors used to detect obstacles.

With Bug1 (Figure 2-18), the robot travels straight to the goal. If an obstacle is reached, a hit-point is registered where after the robot fully circles the robot until the hit-point is reached again. After reaching the hit-point, the robot goes back to the point which was closest to the goal and again starts to travel straight to the goal. This process is repeated until the goal is reached.

Bug2 (Figure 2-18) draws an imaginary straight line from the start to the goal position. The robot travels to the goal along this line until an obstacle is reached. It now circles the obstacle until it reaches the imaginary line. When this line is reached, the robot stops the circumnavigation and follows the line. This process is repeated until the goal is reached.

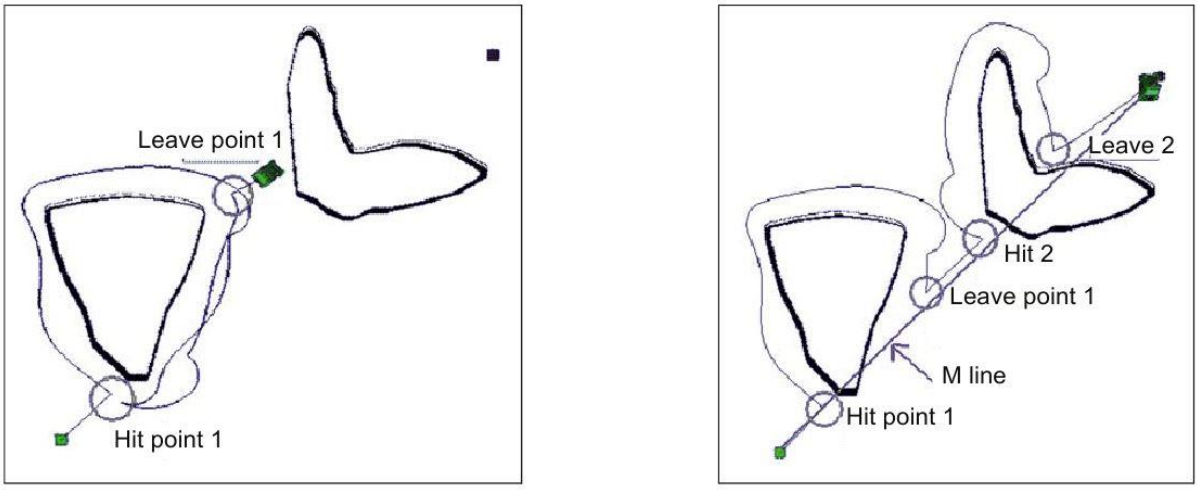


Figure 2-18: Bug1 and Bug2 examples

DistBug or Tangent Bug (Figure 2-19) is based on concepts from the Wandering Standpoint Algorithm. The robot starts by going straight to the goal until an obstacle is reached. It then circumnavigates the obstacle while checking the shortest distance to the goal or whether there is enough free space towards the goal. If there is, then the robot attempts to reach the goal by going straight to it. If it reaches its hit-point, the assumption is that there is no path towards the goal [25].

Although the Wandering Standpoint and Bug algorithms are easy to implement, they suffer from a robustness in local sensors used to determine distances to obstacles and local sensors used to do robot positioning.

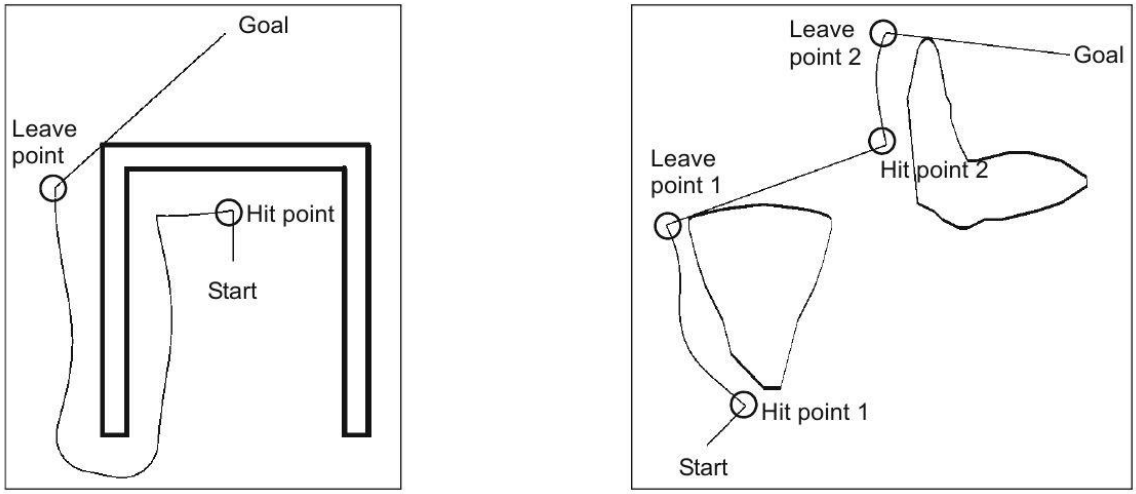


Figure 2-19: DistBug examples

A method that assimilates obstacle avoidance to a known physical analogue is the Potential Fields Method, or according to Jones, Motor Schema [39]. This method combines a homing behaviour with an avoid behaviour. The homing behaviour's purpose is to move the robot to the goal by creating a vector field pointing to the goal from every position on the map (Figure 2-20b). This vector field is created by ignoring all the obstacles on the map. The avoid behaviour creates vectors pointing away from all the obstacles on the map (Figure 2-20c). When these two vector fields are combined, a movement vector for the robot is created based on "pushing" and "pulling" forces (Figure 2-20c). This movement vector will steer the robot towards the goal, while avoiding obstacles.

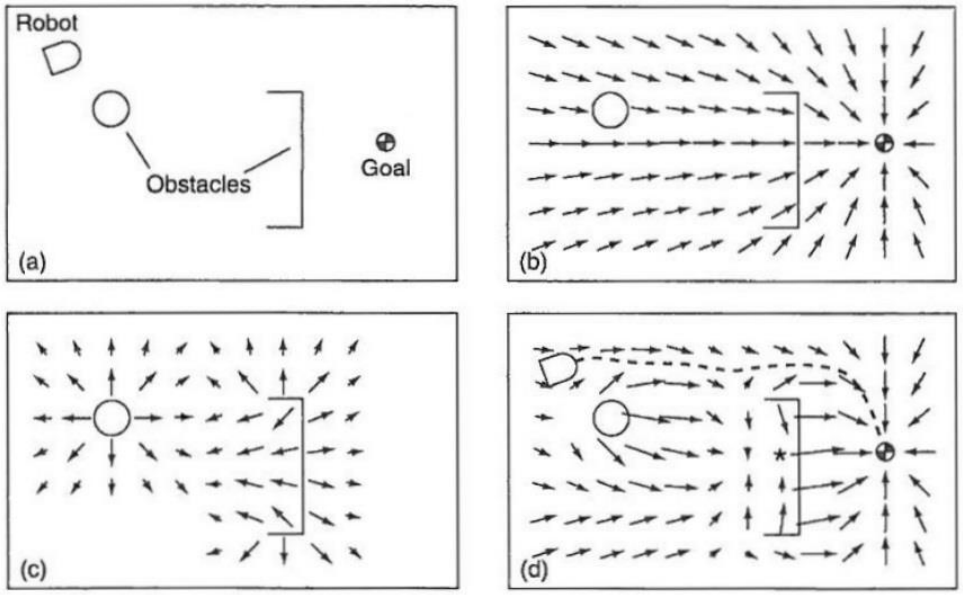


Figure 2-20: Motor Schema

One major problem associated with Motor Schema or Potential Fields, is that of local minima. The local minima is defined by Jones as "a point in the summed vector fields where the robot can become trapped" [39]. This point is where the attracting and repelling forces cancel each other out, where there is zero force [25]. This problem can be overcome by starting a wall following algorithm when progress to the goal has halted or by altering the forces to get the robot moving again. Another less serious problem is that of not necessarily having the global vector field due to incomplete world information. To mitigate this problem, immediate sensor data are used to create the movement vector. Detected obstacles create a repulsive force, pushing the robot away.

This section discussed global and local localisation techniques. It explored node creation methods and highlighted the advantages and disadvantages of the different methods. Path planning was discussed using the created distance graph. Local localisation techniques were compared showing the usage areas of these techniques.

In the following section, the AMR platform and sensor skirt are discussed.

2.3.3 Platform and Sensors

When the map is built and the path to the goal determined, the AMR must still physically move to the goal. This section looks at different designs and design considerations when building the AMR platform. It also discusses sensors and sensor categories as an overview of the sensors used on the AMR.

2.3.3.1 Mobile Platform

According to [38], many different locomotion mechanisms are used by biological systems but due to inefficiencies introduced by similarly scaled man-made systems, mobile robots generally make use of a wheeled mechanism or a small number of articulated legs.

Legged motion requires higher degrees of freedom and greater mechanical complexity but suffers less in rough terrain and on softer ground than wheeled locomotion.

Wheels lose efficiency when moving in soft ground or over rough terrain but are highly efficient on flat, hard surfaces, like those found in a library. Wheels are also generally regarded as moving faster with less energy and as having a smaller control effort due to their simple mechanical design and reduced stability problem [40].

The wheel purposes on a mobile platform are as follows (Figure 2-21) [25]:

Driven Wheels: These wheels are attached to actuators and the rotational velocity of the wheels can be controlled. In the case of a differential drive configuration, individual wheels' velocities can be controlled.

Passive Wheels: Used to ensure proper balance of the platform and can also include caster wheels or unpowered omnidirectional wheels like spherical or Swedish wheels.

Steered Wheels: Wheels can have their orientation with regard to the platform changed by means of a mechanical system, like a steering rack or other actuator.

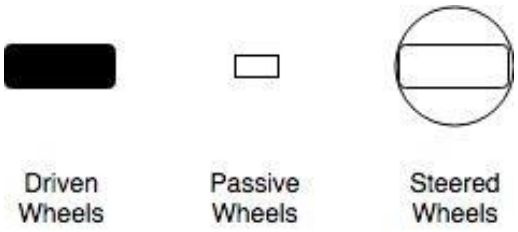


Figure 2-21: Wheel Icons

Although various different wheeled platform designs exist, Gussu and Lin describes the two-wheel differential drive robot as the most popular design [40]. This design consists of two driven wheels with fixed steering axis and one passive caster wheel to facilitate stability even when the platform is static.

Figure 2-22 shows the wheel configuration of a differential drive AMR chassis. It also graphically shows the outcome of the motion based on the individual wheel velocities.

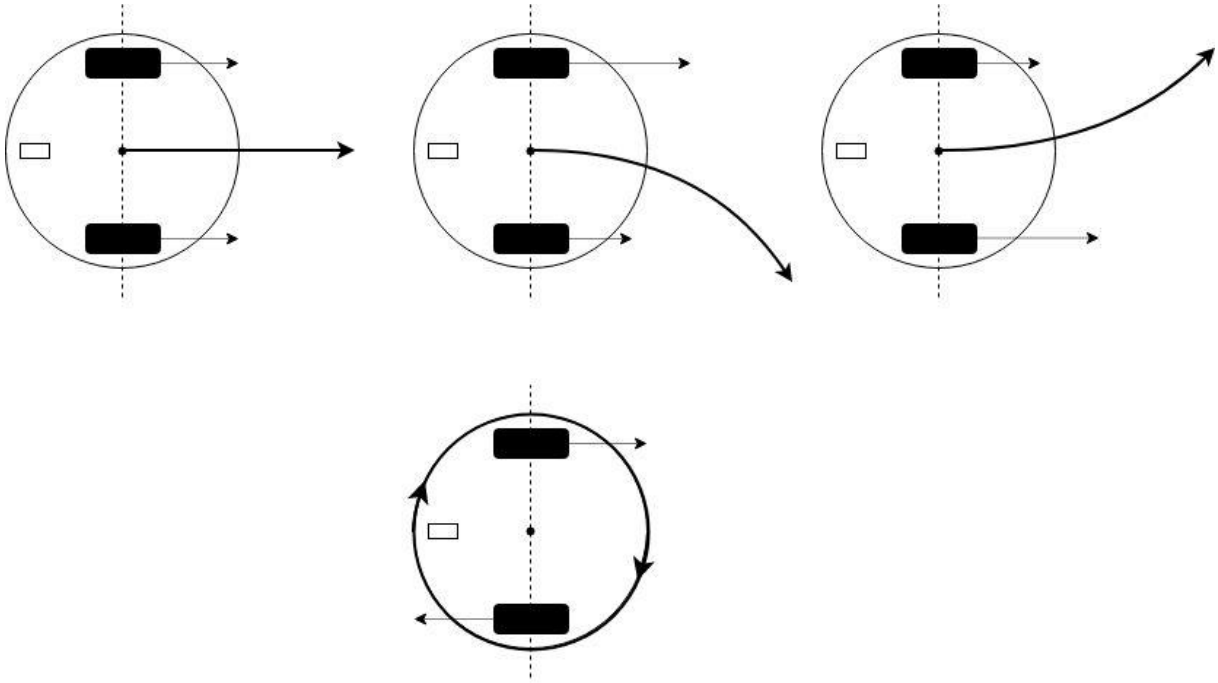


Figure 2-22: Two-Wheel Differential Drive Robot

Differential drive motion is achieved by changing the velocities of the wheels according to the rules in Table 2-2 (Adapted from [25]).

Table 2-2: Differential Drive Movement Rules

Moving the Platform		
Action	**Implementation**	**Condition**
Driving straight, forward	$v_L = v_R$	$v_L > 0$
Driving in a right curve	$v_L > v_R$	$v_L > 0$
Driving in a left curve	$v_L < v_R$	$v_R > 0$
Turning on the spot, clockwise	$v_L = -v_R$	$v_L > 0$

Table 2-3 lists the advantages and disadvantages of a differential drive robot [40].

Table 2-3: Two-Wheel Differential Drive Robot

Two-Wheel Differential-Drive Robot	
Advantages	**Disadvantages**
Simple mechanical structure	Difficulty of moving over uneven terrain
Simple kinematic model	Only bidirectional movement is available
Low fabrication cost	
Zero turning radius	
Systematic errors are easy to calibrate	

2.3.3.2 Sensors Classification and Use

Sensors are paramount to the successful operation of any mobile robot. Measurements of various sensors must be taken to extract meaningful information about the environment and of the system.

Both Bräunl and Siegwart, Nourbakhsh and Scaramuzza classify sensors according to the following important categories [25] [38]:

Proprioceptive or Internal: These are sensors that measure the internal state of the system. For example: motor speed, wheel load, heading, etc.

Exteroceptive or External: These sensors acquire information about the system's environment. For example: light intensity, distance to obstacles, etc.

Passive: Measurement of ambient environmental energy entering the sensor. For example: Microphones, Light-Dependant-Resistors, etc.

Active: Energy is emitted into the environment and the environmental reaction is measured. For example: Sonar sensors, LIDAR sensors, etc.

Local: This group of sensors are physically mounted on the system or robot. For example: LIDAR, Inertial Measurement Units, etc.

Global: Sensors that are mounted in the environment and are used by the robot. For example: Beacons, cameras, etc.

Sensors can further be grouped according to their uses [39]. These uses are:

Collision Detection Sensors: This group of sensors make use of the force between the robot and environment. The force between the robot and the environment is zero if there is no collision. When a collision occurs, the robot provides the force experienced by these sensors through the torque supplied by the driving wheels. Sensors in this category include: Bumper switches, stall sensors and stasis sensors.

Avoidance Sensors: These sensors will try to detect obstacles while they are still a distance away. The sensor data can then be used to avoid obstacles before a collision occurs. Examples include: LIDAR, Infrared proximity sensors, Infrared range sensors and sonar sensors.

Homing Sensors: The purpose of homing sensors is to provide the robot with a means to reach a destination or to be able to determine if a destination has been reached. Sensors in this group include: Photocells, Phototransistors and Photodiodes, coded beacons, pyroelectric sensors, colour blob sensors and magnetic sensors.

Dead Reckoning and Navigation Sensors: The sensors in this category are used to drive the robot to some location without an explicit marker on the place where the robot is going. Examples of sensors in this group include: Shaft encoders, inertial sensors, compasses and GPS systems.

The concept of graceful degradation is where sensors from different categories must be used together to create redundancy and to ensure safe robot operation even in the event of sensor failure and under different environmental circumstances [39].

To this extent, Gussu and Lin introduces the term: Sensor Skirt [41]. A Sensor Skirt is all the sensors providing obstacle detection coverage around the circumference of a mobile robot. The purpose of this sensor skirt is to provide collision-free autonomous movement for the robot to achieve its goal. The sensor skirt should consist of sensors implementing at least two of the different detection methods discussed.

Obstacle detection and avoidance plays a major part in any mobile robot and therefore sensors from this group will be examined further.

Active ranging is when energy is emitted into the environment and the response of the environment is measured. These types of sensors continue to be the most popular sensors in mobile technology since many sensors have a low price point with easily interpreted outputs.

Two basic methods used by active ranging sensors are: 1) time-of-flight, and 2) geometric methods.

Time-of-flight ranging makes use of the propagation speed of sound or an electromagnetic wave like light or radar. The distance to an obstacle is calculated by taking the return time of an emitted pulse and multiplying that time by the speed of the wave propagation. Since the time taken is from the transmitter to the obstacle and back, the distance is a round trip distance and must be divided by 2 to determine the actual distance to the obstacle.

Typical time-of-flight sensors are ultrasonic sensors. Ultrasonic sensors are appealing in terms of size, price and power consumption [29]. They operate by transmitting a series of sound pulses. These sound pulses travel through the air at a speed of approximately 343 m/s, depending on temperature and altitude. After the series of sound pulses are transmitted, a timer is started and a threshold value is set to determine if an incoming sound wave is a valid echo. During the initial transmission, this threshold is set very high to suppress triggering the echo detector with the outgoing pulses. This period where the threshold is very high is known as the blanking time of the sensor and in this period, the sensor is not able to detect any incoming echoes. Ultrasonic waves typically have a frequency of between 40 kHz and 250 kHz with a detection distance of between 120 mm and 5 m. The accuracy of these sensors is reported as between 98% and 99.1% and can achieve a resolution of approximately 20 mm [38][42].

Depending on the manufacturer, these sensors can have an acoustical cone that varies between 20° and 40°. Measurements are repeated at a rate of about 20 times per second for a frequency of 50 Hz, which is a relatively slow cycle time. Implementing more sensors can have a detrimental effect even as each sensor added will increase cycle time for the sensors. This can negatively influence even robots conducting moderate speed motion since the time between sensor data is increased.

In order to speed up acquisition times, some researchers propose to simultaneously employ sonar sensors on opposite sides of sonar rings [25][38].

Unfortunately, ultrasonic sensors are not free from limitations [38] [39] [43].

- Poor directionality limits the accuracy in detecting an edge based on the angle between the obstacle surface and the acoustic beam (Figure 2-23a).

- Specular reflections, which occur when the angle between the acoustical beam and the surface is so large that the incoming ultra-sound is reflected away from the sensor (Figure 2-23c).
- The obstacle is either not detected or seen smaller than it is due to a lack of reflected energy (Figure 2-23b).
- Stray reflections from neighbouring sensors induce crosstalk which leads to errors in the distance data.

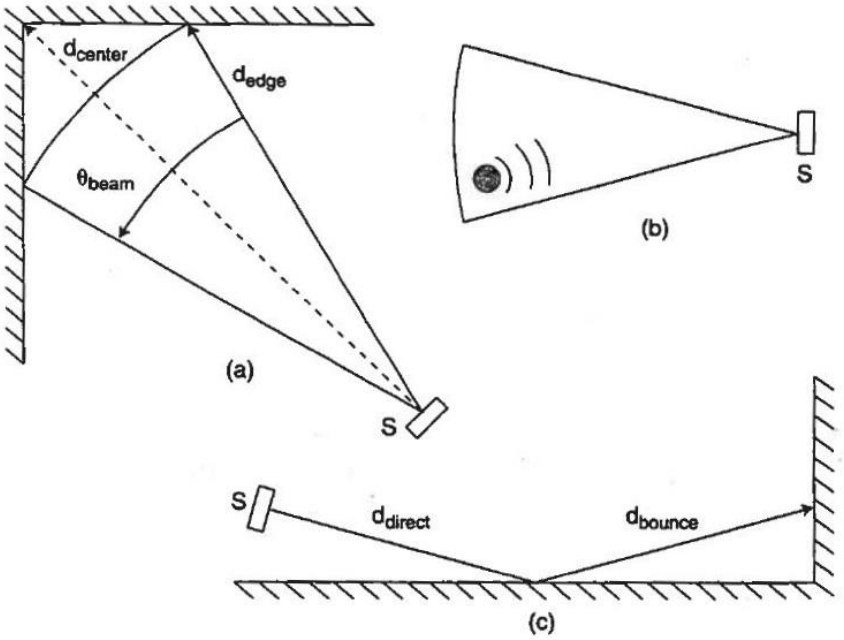


Figure 2-23: Ultrasonic Limitations

Active triangulation ranging is part of the Geometric methods of distance detection. 1D optical triangulation makes use of a collimated beam of infrared light. The reflected light is focused onto a position-sensitive-device (PSD) and the distance is calculated by using the angle of the reflected beam.

Structured light is another method of active triangulation that is used. These rangers project a known light pattern, structured light, onto the environment. The receiver captures this light pattern and by comparing it to the known geometric pattern, simple triangulation between

the known pattern and captured pattern can be used to determine distance values (Figure 2-24)[44].

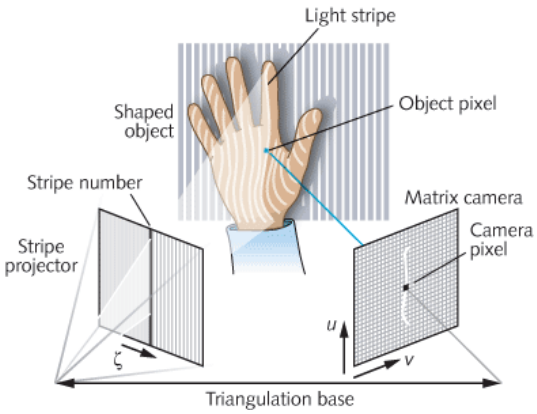


Figure 2-24: Structured Light Sensing Principle

Figure 2-25 [45] shows the geometric pattern from a Microsoft XBOX 360 Kinect V1 Sensor projected onto a person in front of a wall. This shadow seen in the photo is due to the slight offset between the projected IR laser pattern and the camera.

This geometric pattern is reflected from surfaces and read by the IR camera. Objects in the Field of View (FOV) of the XBOX 360 Kinect V1 sensor, distort the original dot pattern. The amount of distortion is determined by the distance of the obstacle from the IR camera. The closer the object to the camera, the smaller the difference between the original dot pattern and detected dot pattern will be. Using these differences, a height map or distance map of obstacles in the FOV of the XBOX 360 Kinect V1 sensor is generated.

Figure 2-25: XBOX 360 Kinect V1 Infrared Dot Pattern

Although additional light sources might interfere with structured light sources [46], a major advantage is that since structured light sensors are active devices, it will continue to work in low light or even dark environments. Passive image analysis might fail in the same low light environments and will not work in completely dark environments.

It is important to note that sensors making use of light, in this case infrared light, are subject to the normal rules that apply to visible light. Sensors using light cannot detect glass or transparent plastics since these materials refract light. When light is refracted, the sensor is unable to determine distance and according to the sensor, the obstacle does not exist.

2.4 Robotic Software and Firmware Architecture

National Instruments (NI) describes robot software architectures as sets of hierarchical control loops [47]. These loops are presented in layers and in turn represent high-level planning on high-end computing systems through layers controlling path planning, obstacle avoidance and other tasks, down to low-level actuator and sensor control. This layered approach from NI, is shown in Figure 2-26.

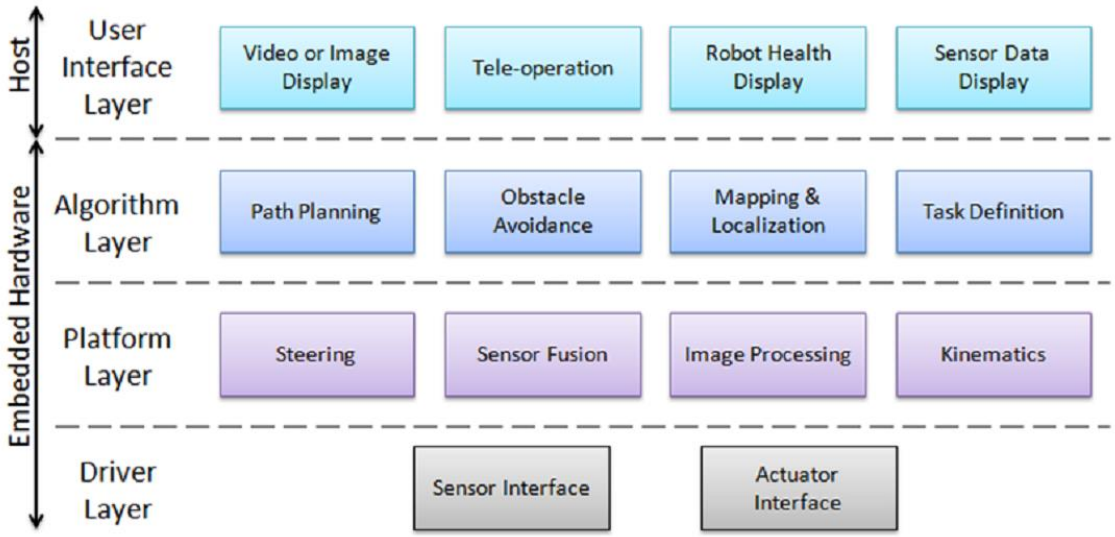


Figure 2-26: Robotics Reference Architecture

2.4.1 Driver Layer (DL)

The driver layer handles the low-level driver functions required to operate the robot. It takes raw sensor data and turns it into meaningful engineering units like position, speed, etc. This layer is also responsible for taking motor velocity values and converting them into low-level signals which are sent to the motor controllers.

By changing the driver layer, developers can switch between actual hardware and a simulation.

2.4.2 Platform Layer (PL)

This layer contains the physical hardware configuration of the robot and it converts data from the path planning algorithms to actuator data, which is then sent to the motors on the driver layer.

It also combines sensor data and sends this newly combined data to the algorithm layer for higher level path planning and localisation.

2.4.3 Algorithm Layer (AL)

Functions in this layer represent the high-level control algorithms and contain high-level path planning and mapping algorithms which will use system information such as velocity and position data to make control decisions.

2.4.4 User Interface Layer (UIL)

The UIL's purpose is to provide meaningful feedback to users. It can make use of telemetry and/or teleoperated functions controlled by keyboard, mouse or joystick input.

The UIL is not fully implemented since the robot's goal is determined by the book being placed on the platform and not directly selected by human operators.

2.5 Chapter Summary

In this chapter, the current research into library automation is highlighted and it was shown that the focus is on: 1) Ways of improving asset management by implementing RF scanners to look for and identify books, or 2) Directly influencing the user experience in assisting users to find specified books. It was noted that little research has been conducted in getting books to their right places in the shelves after being returned by patrons/users.

This chapter also answers the questions: “Where Am I?”, “Where Am I Going?” and “How Should I Get There?” which were discussed in this section. The focus was on map creation techniques, different localisation techniques, effective path planning algorithms and the type of platform to use and which sensors are most appropriate for the research application.

A motivation for wheeled robots is made highlighting the differential drive configuration as an efficient and stable platform to use in environments like libraries.

This section further discusses sensor classes and categories, focussing on active ranging sensors by discussing their advantages and disadvantages.

This chapter concludes by highlighting the hierarchical control loop design structure when creating control software and firmware for mobile robots.

The following chapter will focus on the research methods followed to complete this study.

Chapter 3 – Development Process of Automated Book Truck

This chapter introduces and discusses the theories and methodologies comprising the core of the study.

To prove the hypothesis, the following research strategy was implemented:

- Design and Creation Research strategy is a problem-solving approach where the following iterative steps are used: Awareness, Suggestion, Development, Evaluation and Conclusion [48].

3.1 Planned Design Strategy

In the development of the simulator and the AMR platform, the Design and Creation research strategy was used to select the most appropriate algorithms and concepts identified in the literature study.

This strategy consists of the following five iterative steps:

- Awareness – Conducting a literature study to become familiar with each research sub-question.
- Suggestion – A possible solution is identified on how to answer each research sub-question.
- Development – Implementing the proposed solution for each research sub-question.
- Evaluation – Examining the implemented solution to determine its worth.
- Conclusion – Consolidating the evaluated results to select the best solution for each research sub-question.

The process of working through the stages of the Design and Creation research method is known as the systems development method. This process should not be confused with the methodology of the study, which combines all the research strategies and data gathering methods used in the research [48].

3.1.1 AMR Platform Design Considerations

According to the literature reviewed, all the studied AMRs had about the same footprint size as a human library user. Many of the studied AMRs also have the capability of interacting with books, either by having the books placed on dedicated shelves on the AMR or by having the books presented to scanners mounted on the AMR.

The designed prototype will also have the same footprint as a human library user. However, since book manipulation, where the AMR might move or interact with a book, is not part of this research effort, book manipulation is omitted from the prototype. Creating a prototype with a human sized footprint will not only allow book interaction to be added at a later stage but will also provide more realistic test results.

Motion will be achieved by implementing a differential drive system. As shown, this is an efficient and simple solution especially in an environment like a library where the floor surface is hard and flat.

3.1.2 Design Considerations for Sensor Selection

The successful use of sonar sensors in various researched localisation applications is shown in the literature study. Sonar sensors have also been used as obstacle avoidance sensors. However, due to the lack of high-resolution data, sonar sensors will be used as the primary sensors for the localisation algorithm and only have obstacle avoidance as a secondary function.

Since localisation will be done with the measured distance between the ultrasonic sensors and the books in the shelves, this very lack of high-resolution data will be used to filter out distances to books which might not be neatly placed back into the shelves. Exploiting the

noise of the sensor enables distance data to essentially be “pre-processed” and frees up the controller from doing distance averaging on the sensor data.

Map updates, when obstacles are detected, will be done by using data from the XBOX 360 Kinect V1 sensor. The data provided by the XBOX 360 Kinect V1 sensor will be used for obstacle avoidance by implementing vector fields.

3.1.3 Mapping and Localisation Considerations

The decision was made to implement a combination of mapping methods as the solution. The occupancy grid map will be used to record obstacles while the continuous space map will be used to record the AMR position.

Due to noisy sensors and therefore inaccurate pose data, localisation will be implemented using probabilistic methods. Additionally, since AMR movement is recorded on the continuous space map, Particle filters (Monte Carlo Localisation) will be used to localise, as opposed to Markov Localisation which is a grid-based approach to localisation.

Odometry data from the AMR will be fused with the Particle filters to complement the localisation process.

3.1.4 Library Remodelling

A big consideration is the implementation of the AMR into a traditional library without needing to remodel the library. This very reason is why it was decided to not implement SLAM but rather provide the robot with a map. Due to this given map and the ability for the AMR to localise itself, no markings or signage in the library is necessary since all the information is on the provided map.

Libraries consist of aisles with various widths and although research has been done using sonar sensors to move in narrow aisles, it was decided to limit the movement of the AMR to the wider aisles. This will not only facilitate quicker movement to the drop-off points but will also limit potential collisions due to narrow aisles.

3.2 Chapter Summary

This chapter discusses the methodology selected for this study using the discussed theories and methodology to create a research plan.

Exploratory research was used to implement existing knowledge regarding localisation, mapping and sensors in a library environment.

Design and creation research was used as a strategy to select the most appropriate method from the existing literature. The chosen methods were used to build the prototype and answer the research question.

Evaluation research was used to evaluate the suitability of the proposed solution to answer the research question. These results are shown and discussed in Chapter 5 – Design Implementation.

Chapter 4 – Algorithm Layer and Hardware Design

Chapter 4 describes the implementation of an automated book truck in two sections: 1) The software implementation of the sub-systems needed to create a simulation of the autonomous mobile robot as a first step to answer the research question, 2) Introduction and explanation of the suggested physical platform to be used to implement the automated book truck.

4.1 Algorithm Layer Design

Many robotic simulators already exist. Examples include: ROSS, LabView Robotics, Microsoft Robotics Developer Studio, and Player/Stage, to name but a few.

Simulation systems allow designers to understand, develop, test and improve algorithms before implementing them on actual built hardware systems. This chapter introduces the simulation developed for this study, which also doubles as the Algorithm Layer used to control the robot hardware.

4.1.1 Software Design of the Algorithm Layer

For this study, the control loops used to implement an Algorithm Layer (AL) and User Interface Level (UIL), as discussed in section 2.4, are described.

The language used was “Processing”, which is a Java wrapper language that can run on Windows, Mac and Linux operating systems without complicated installation procedures. The simulator was built from the ground up to keep software installation processes as simple as possible.

Inspiration was taken from the already existing robotic software and a simulator was developed that would be capable of interacting and interfacing with the actual hardware of the robotic platform at a later stage.

The following steps briefly highlight the sequence of events the Algorithm Layer executes in order to move the robot from the starting position to the goal and back:

1. When the simulator starts, the provided map file will be loaded and converted into an occupancy grid map creating fixed obstacles.
2. The AL will attempt to localise the robot using data from the sonar sensors.
3. The goal position will be determined depending on the path selected.
4. An optimum path is planned towards the goal.
5. Movement data are then sent to the driver layer.
6. The map is updated based on data received from the sonar sensors and from the XBOX 360 Kinect V1 sensor.
7. If the map changed, a new node list must be created for the path planning algorithm.
8. Repeat steps 2 to 7.

Steps 2 to 7 are repeated continuously until the robot returns to its original starting position.

Figure 4-1 shows a flowchart depicting the above steps graphically.

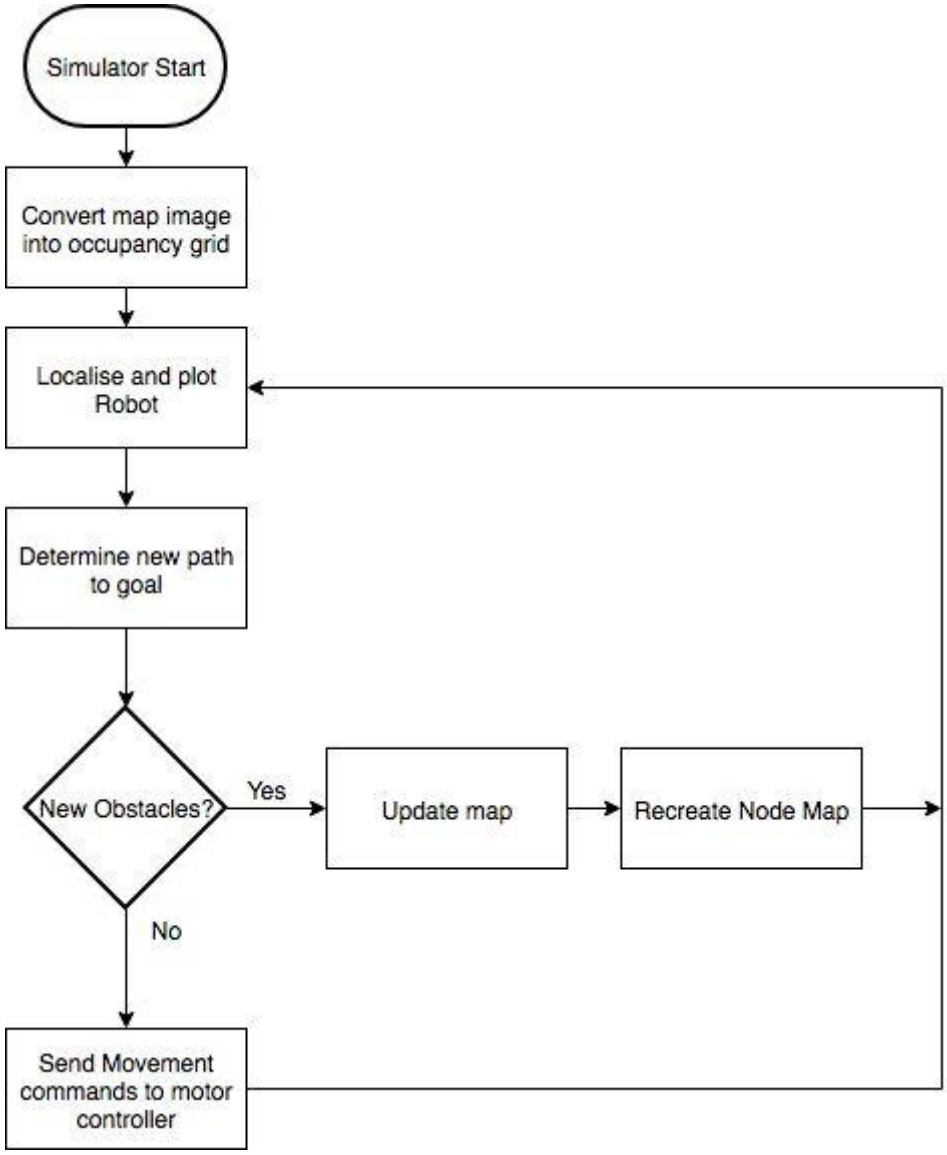


Figure 4-1: Algorithm Layer Flowchart

4.1.1.1 World and Route Map

The simulator makes use of an a priori map. Consequently, the map needs to be available and must be imported into the simulator.

To create the map, a scanned or CAD drawn floor plan image of the area must be available. This image must be edited to remove all unnecessary text and non-relevant markings. Fixed obstacles, such as desks, pot plants, turnstiles, walls and restricted areas, can now be added by marking these areas as black.

When the simulator starts, the processed image (Figure 4-2) is loaded from the saved PNG or JPG file. The world map (Figure 4-3) is created using the loaded image by converting it into an occupancy grid map where the walls, obstacles and restricted areas are indicated on the occupancy grid map as RED, while GREEN areas indicate places the robot can travel.

Figure 4-2 shows a small room map after all unnecessary artefacts have been removed and Figure 4-3 shows the occupancy grid map after the AL has started and converted the original map.

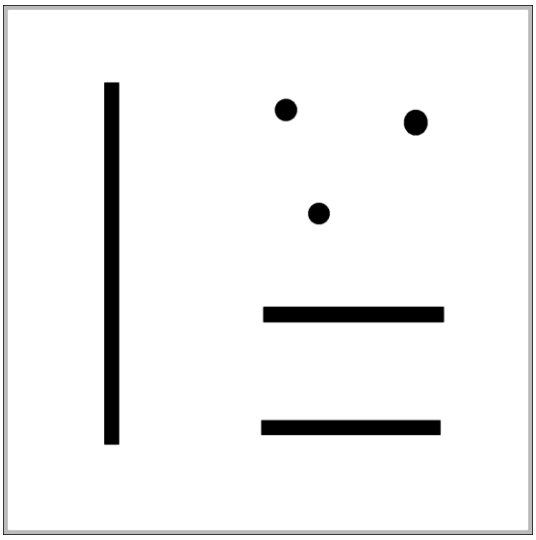


Figure 4-2: Floor plan

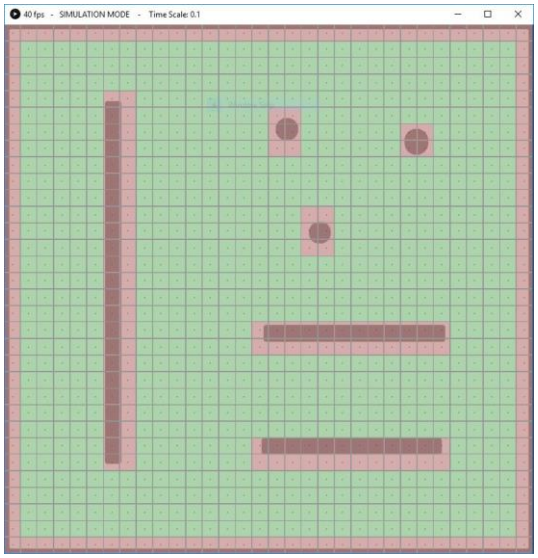


Figure 4-3: Occupancy Grid Map

Although the obstacles are plotted into an occupancy grid map, the robot is not limited to only moving in the centre of the occupancy grid blocks. It can move freely throughout the map, except on occupied blocks.

To generate the grid map representation, the floor plan image is scanned from left to right and top to bottom to determine if a pixel represents an obstacle. If the pixel is black, the grid block associated with that pixel is identified and the tile type is set to MAP. This process is repeated until the image is completely scanned (Figure 4-4) (Listing A-1).

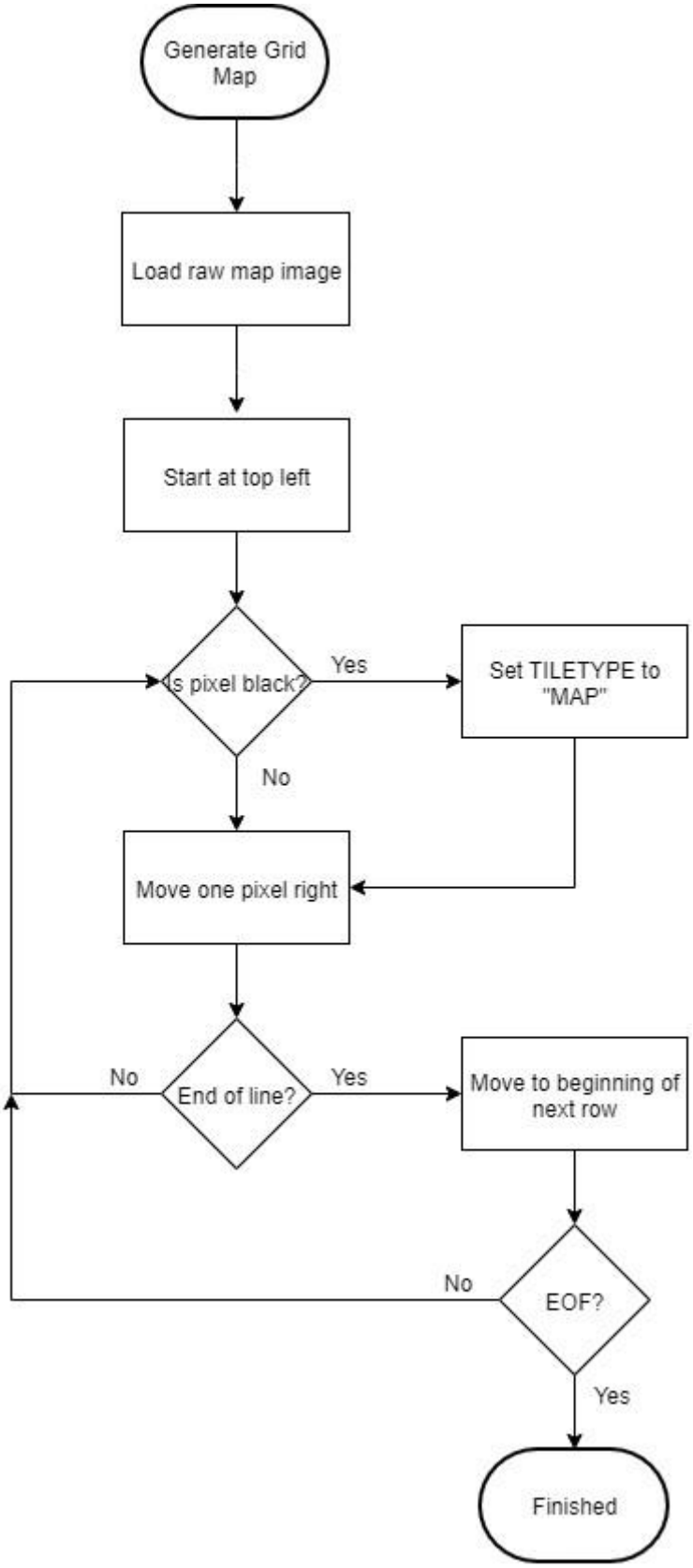


Figure 4-4: Grid Map Creation Flowchart

4.1.1.2 Localisation

Two data sets are used to determine the location of the robot. The first data set for the location is determined by using dead reckoning. As described in previous chapters, dead reckoning is a method used to determine position by starting at a known location and adding small linear increments to the starting location, based on the kinematic model and wheel speeds of the robot. The second set of data is determined by implementing particle filters. Both these sets of data are combined to determine the location of the robot in the world map.

Each particle consists of a 'nose' indicating the direction the particle is facing in and a red ellipse as the base of the particle (Figure 4-5).

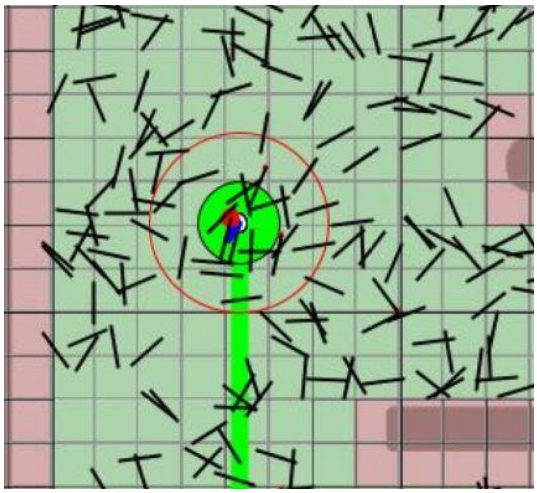


Figure 4-5: Red Ellipses indicating probability value

The size of this ellipse is in relation with the probability match of this specific particle. The bigger the probability match, the bigger the red ellipse. On closer inspection, most of the particles do not have a red ellipse, meaning that they are not probable contenders for the position of the robot. Close to the robot, a few particles can be seen with red ellipses.

Figure 4-6 shows an initial particle distribution of 1000 particles on a map of a fictitious room.

After the first scan-and-move cycle (Figure 4-7), the particles are grouped based on the probability matches of their simulated sensors and the data from the robot's sensors. Multiple groups exist due to symmetry in the map.

As the program continues to execute, localisation is refined until it reaches cycle 7 (Figure 4-11), where the probability mass of the particles has synchronised with the robot and localisation is achieved. Listing A-2 contains two functions used to calculate the probability value of each particle. The sense() function is used to calculate the distance values for each sensor on the particle. This distance data are then used by the measureProb() function to calculate the fitness of each particle.

Although the sequence of screenshots (Figure 4-6 to Figure 4-11) shows that particle filters and odometry can achieve localisation from a completely unknown position, the robot was started from a known position before invoking the program. Particle spawning was limited to an area surrounding the robot. This eliminated the possibility that the particles might find an additional position favourable due to symmetry in the environment. The starting point of the robot is part of the calibration and start-up procedures of the robot.

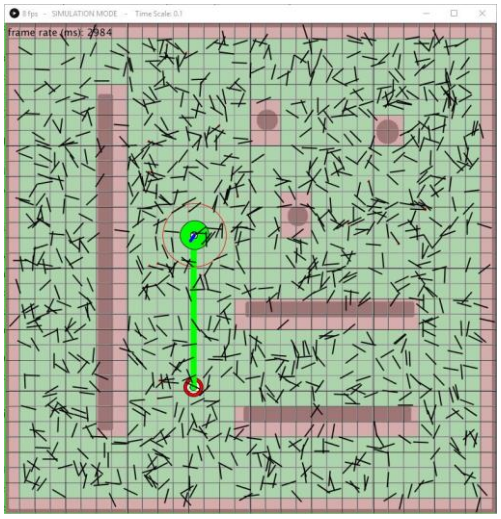


Figure 4-6: Initial Particle Distribution

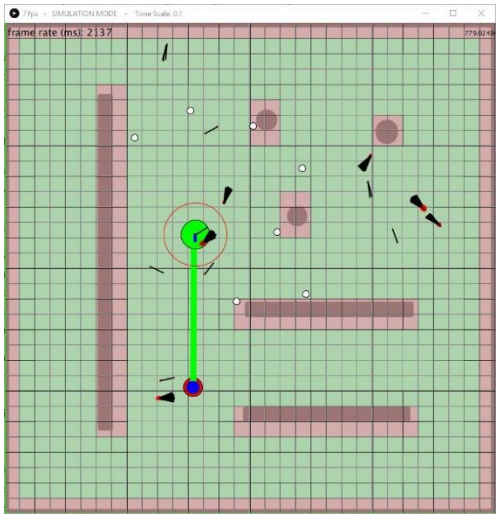


Figure 4-7: Distribution after 1 cycle

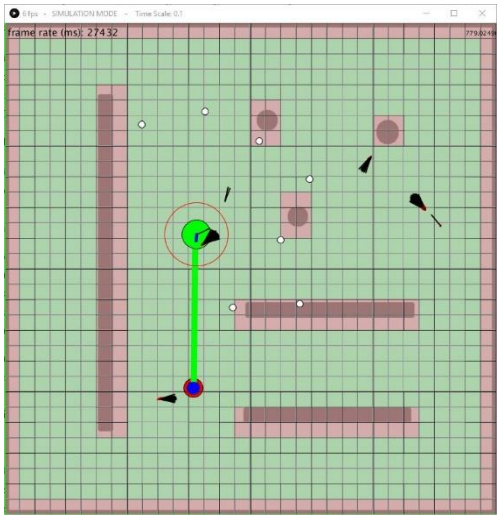


Figure 4-8: Distribution after 2 cycles

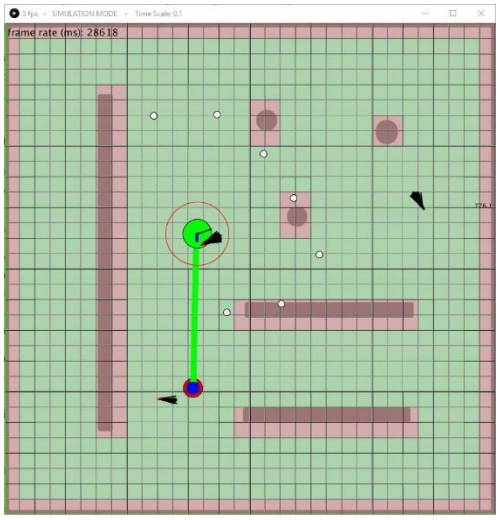


Figure 4-9: Distribution after 4 cycles

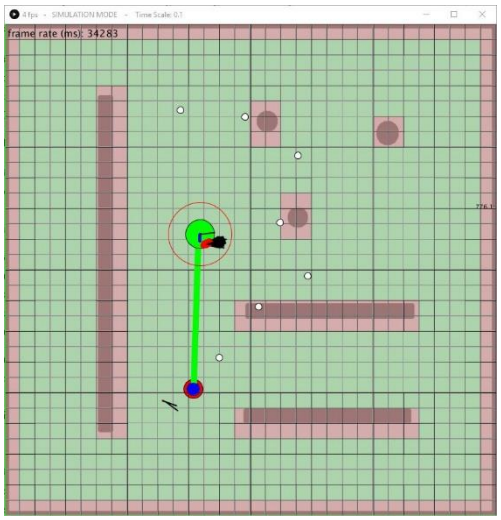


Figure 4-10: Distribution after 5 cycles

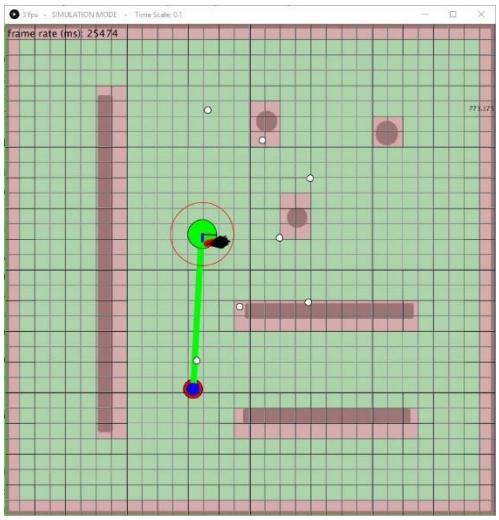


Figure 4-11: Distribution after 7 cycles

4.1.1.3 Path Planning

The AL constantly calculates the best path from the robot to the goal as described using the A-star method. The nodes identified by the A-star method are saved into a list and the robot must go to each of these nodes in sequence to reach the goal.

As the robot moves through the environment, its position changes and therefore a shorter, more optimised route might become available. The current path might also be influenced by newly discovered obstacles as they are being detected by the sensors.

Path planning is done on 2 levels: 1) Global path planning and 2) Local path planning. Global path planning makes use of the Quad tree method to create nodes or waypoints the robot must navigate to in order to reach the goal. Local path planning makes use of potential fields which has the robot navigate obstacles in the immediate vicinity of the robot and incorporates immediate obstacle avoidance using the XBOX 360 Kinect V1 and sonar sensors. The combination of these two levels' vector data determines the direction the robot will move in.

Global Path Planning

Path planning is achieved using a node graph and the A-star method. The node graph is generated using the Quad-Tree method of dividing the map.

As described, quad tree node analysis recursively divides the map into smaller blocks (quads) until areas are identified without obstacles or when a specified resolution is reached (Figure 4-12)(Listing A-3). To ensure that the map is dividable to a single block, the maximum number of horizontal and vertical blocks is calculated to be a factor of 2^n. This might change the defined grid size to achieve the desired division factor.

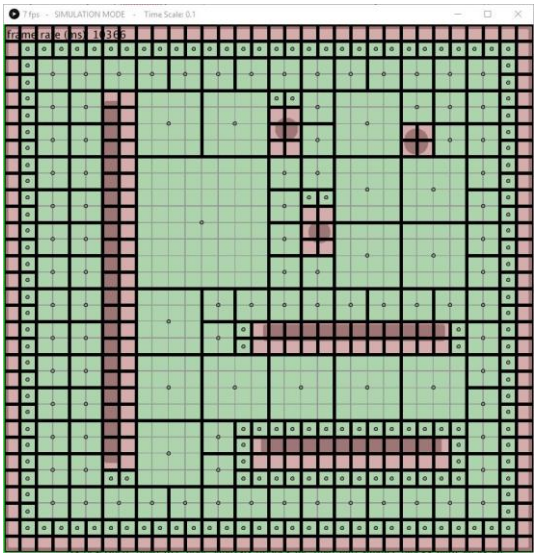


Figure 4-12: Quadtree Analysis of map

Whenever the map changes, which might be due to user added or sensor added obstacles, a new quad tree analysis is done, creating new nodes which, in turn, are used by the AL to determine the best path to the goal.

Quadtree generation is a recursive process which repeats until all the quads generated are either empty or occupied or when a specific level of division has been reached.

Local Path Planning

Local path planning is achieved using a combination of potential fields and sonar sensor data. Potential fields (Figure 4-14) are generated for each of the block-based pushing and pulling forces experienced by a specific block. Each block has a vector regardless of whether the robot is on the block or not.

A movement vector for the robot is determined by combining the sonar sensor data, the vector to the next waypoint, and the block vectors.

Every time the map updates, the pushing force of the tiles is updated, and the potential fields used by the robot change. The obstacles generate a pushing force whereas the goal creates a pulling force. Combining these forces generates the vector which the robot follows towards the next waypoint or the goal (Listing A-4).

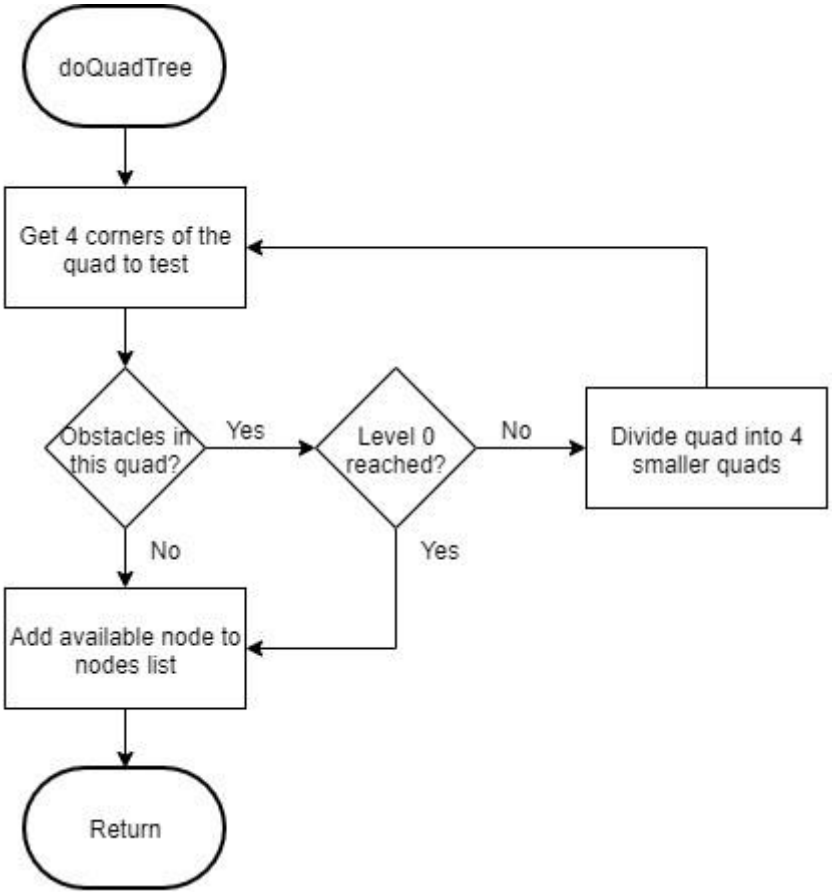


Figure 4-13: Quadtree generation flowchart

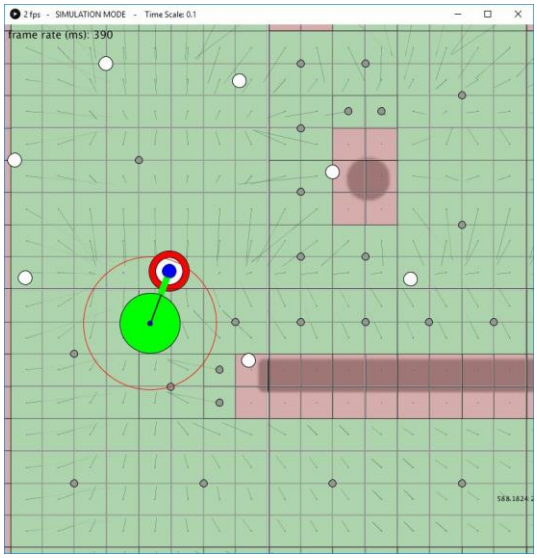


Figure 4-14: Potential Fields

Obstacle Avoidance

Obstacles are detected using both the sonar sensors and the XBOX 360 Kinect V1 sensor. Although the sonar sensors' primary purpose is to localise, the data are also used for obstacle avoidance if distances to detected obstacles fall below a minimum threshold. The sonar sensor data will not update the map and is only used to avoid obstacles and to localise.

Data from the XBOX 360 Kinect V1 sensor updates the map which in turn is used by the Localiser and Path planner to determine the best route for the robot to take. Figure 4-15 shows the data flow model of the Algorithm Layer and which process the sensor data influences.

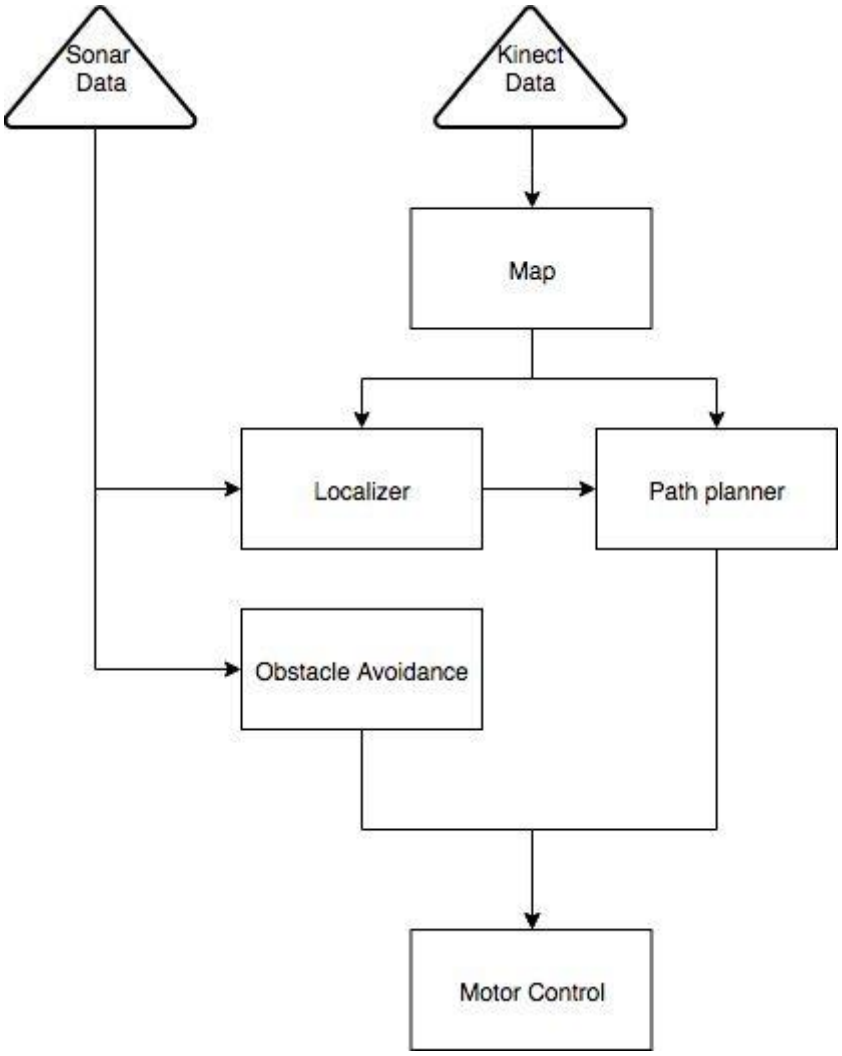


Figure 4-15: Data Flow Model

All obstacles have the following properties: A gravity value, and a tile type. Initial gravity values are determined by the tile type. Tile types can be one of the following: 1) Unassigned, for tiles that are empty, 2) Map, for tiles containing fixed structures as determined by the initial scanned map, 3) User, for obstacles added by the user while the AL is running, and 4) XBOX 360 Kinect V1, for obstacles detected by the XBOX 360 Kinect V1 sensor.

Table 4-1: Tile Type Gravity Values and Colour

Tile Type	Gravity	Colour
UNASSIGNED	0	Green
MAP	255	Red
USER	255	Blue
XBOX 360 Kinect V1	Variable	

The value of the gravity variable depends on the type of the tile. For MAP or USER type tiles, the gravity will never decrease since walls and user added obstacles are permanent fixtures. Gravity values for the UNASSIGNED tile type are zero. This shows an empty tile, devoid of any obstacles. The XBOX 360 Kinect V1 tile type gravity is a variable value and it depends on how many ‘hits’ the XBOX 360 Kinect V1 sensor had in a specific scanned column.

The physical gravity value does not influence the method or severity as to how the robot avoids obstacles and the robot will treat a tile with a gravity value of 50 the same way as a tile with a gravity value of 250.

Gravity values are used as 1) a method to filter any noise by only mapping an obstacle when the gravity value is above 10 hits and 2) decreasing the gravity value at a constant rate to automatically remove obstacles in order to compensate for the dynamic part of the map where obstacles might move around or be moved around. Point cloud data from the XBOX 360 Kinect V1 sensor is divided into height-wise columns. The amount of points in each column determines the weight of that column and ultimately determines the gravity of the occupancy grid this column represents.

The width of these columns is determined by the grid size selected for the occupancy grid map and stays constant regardless of the distance from the sensor. Fewer columns are closer to the sensor with an increase as the detection distance increases.

The decay rate is a constant value applied to all XBOX 360 Kinect V1 obstacles. A block with more ‘hits’ will thus take longer to disappear since the system was surer that it was an actual obstacle and not just noise from the sensor.

Fixed Obstacles (MAP and USER Tiles)

At the very least, fixed obstacles are the walls of the world. Other structures that can be considered as permanent include the likes of tables and chairs which do not move, pot plants which do not move, and other furniture in fixed positions. Any structure not included in the original map and not seen as fixed obstacles will be detected by the sensors and added to the map. These structures will be treated as non-fixed obstacles.

Users can edit the map after import. These added obstacles are known as USER tiles. The only difference between USER tiles and MAP tiles is that USER tiles can be deleted and moved by a user after the software started, whereas a MAP tile is permanent and can only be changed by changing the scanned floor plan.

Non-Fixed Obstacles (XBOX 360 Kinect V1 Tile)

This category consists of all structures not included in the original a priori map and structures which are not fixed, for example, people, chairs, and other structures. These structures are placed on the map whenever they are detected by the XBOX 360 Kinect V1 sensors.

A decay rate is introduced which allows these obstacles to be removed from the map after the decay period elapsed. Since they are non-permanent, the assumption is that these obstacles might move around and therefore ‘disappear’ after a certain unknown period of time.

Other library users are part of this group since users might be in the way of the robot while browsing a book and after finding the book they might move to another area of the library.

The decay rate simulates this unknown time and removes XBOX 360 Kinect V1 tiles to allow the path planner to use that space for path planning again.

4.2 AMR Design and Implementation

This section looks at the physical implementation and building of an AMR according to the literature studied in Chapter 2 and further explains why certain choices were made.

4.2.1 AMR Platform

The mobile robot platform used in this study is the MadeUSA/Arlo mobile platform. The MadeUSA/Arlo mobile platform is distributed by Parallax Inc. and consists of the Robot Base Kit (#28976, #28977) [50].

This platform was selected because it has a diameter of 460 mm and therefore:

- Is large enough to enable it to carry books.
- Has a footprint very close to the same size as a human's footprint and will therefore be able to handle the isles between shelves in a manner similar to humans.
- The height, currently at 500 mm but adjustable to 800 mm, and size of the platform makes it very visible when moving in a human occupied environment and therefore helps to reduce possible collisions between humans and the AMR.

The MadeUSA/Arlo mobile platform is a differential drive mobile platform. It has the capacity to have two caster wheels added for stability. Although Bräunl suggests to use 2 caster wheels to ensure a stable platform, it was decided to only use one caster wheel fitted to the back of the platform [25]. To ensure platform stability, the centre of gravity of the robot was moved backwards by moving the battery and other control circuitry to the back of the platform. This decision was based on multiple castor wheels standing a chance of beaching the robot when driving on uneven terrain, like what might be found in an environment where there are ramps, carpets or other floor obstacles.

Figure 4-16 shows the automated book truck consisting of the wheels attached to the bottom platform and the ultrasonic sensors attached to the upper platform. The position of the XBOX 360 Kinect V1 sensors can be seen and although a laptop is present on the upper platform, this is temporary, since that space will be used to place the books that must be delivered to the drop-off points.

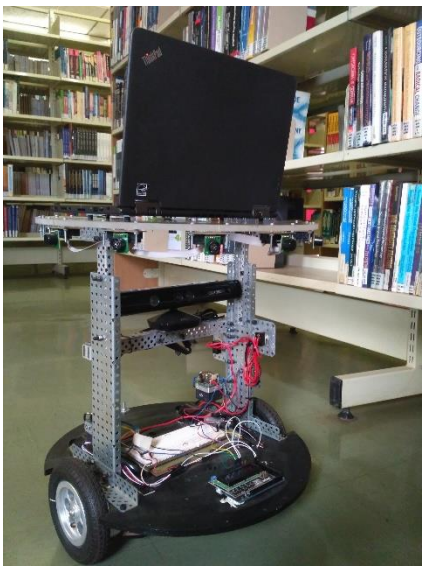


Figure 4-16: Automated Book Truck

4.2.1.1 Wheel Encoders

Attached to the MadeUSA/Arlo mobile platform is the Motor Mount and Wheel Kit (#27971) (Figure 4-17)[51] with Position Controller (#29321)(Figure 4-18)[52].

Figure 4-17: Motor Mount and Wheel Kit

The Motor Mount and Wheel Kit consist of 153 mm pneumatic rubber tires with an attached encoder disk, consisting of 36 fins. These fins, together with the quadrature encoder, provides for a total of 144 ticks per wheel rotation which equates to about 3 mm of linear travel per tick [52].

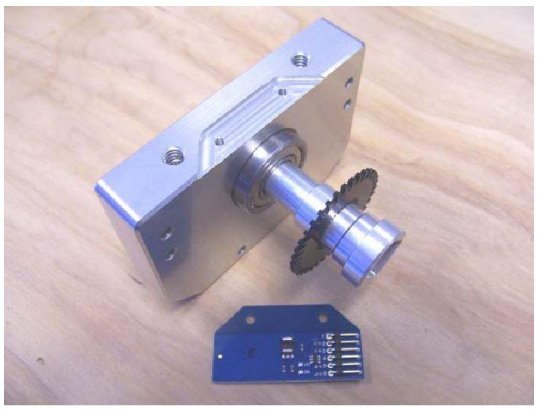


Figure 4-18: Position Controller

Data from the quadrature encoders are read using an Arduino Mega board. Interrupts on the Arduino Mega board are enabled to ensure that no encoder transitions are lost.

The decoding of the quadrature encoder pulse trains is done according to Kellet [53]. Every time a pulse train is sampled, the current state of the A and B channels of the quadrature encoder is combined with the previous state of the A and B channels. This combination creates a 4-bit binary word which is used as an index to determine if a value of +1, 0 or -1 should be added to the running total of the number of ticks (Figure 4-20).

Figure 4-19 shows the combination of the previous and current values of the Channel A and Channel B encoder data. The created decimal index number is used in Figure 4-20 to select the value that must be added to the running total of the number of ticks for each wheel.

old A	old B	A	B	dec
0	0	1	0	2
1	0	1	1	11
1	1	0	1	13
0	1	0	0	4

old A	old B	A	B	dec
1	1	1	0	14
1	0	0	0	8
0	0	0	1	1
0	1	1	1	7

Figure 4-19: 4-Bit Table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	-1	+1	0	+1	0	0	-1	-1	0	0	+1	0	+1	-1	0

Figure 4-20: Encoder Look Up Table

Listing A-5 shows the setup and implementation of Kellet [53] using code compatible with Arduino Mega controllers.

4.2.1.2 Drive Kinematic Equations for Differential Drive Robot

Differential drive robots make use of well-known simple geometric equations [25] [28]. These well-known equations are used to determine the mobile platform's pose after movement and makes use of the interpreted encoder data as previously described.

The inverse kinematic equation was implemented using the following equation [25]:

$$\begin{bmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{bmatrix} = \frac{1}{2\pi r} \begin{bmatrix} 1 & -\frac{d}{2} \\ 1 & \frac{d}{2} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Equation 4-1: Reverse Kinematics

Where:

- $\dot{\theta}_{L,R}$ are the individual wheels speeds measured in revolutions per second.
- d is the distance between the two drive wheels.
- v is the platform's required linear speed.
- ω is the platform's required rotational speed.
- r is the wheel radius.

In order to ensure speed synchronisation between the wheels the schema from Figure 4-21 is implemented [25]. This schema uses PID controllers to control individual wheels' speeds and an Integral controller with curve offset to ensure the wheel speed stays synchronised.

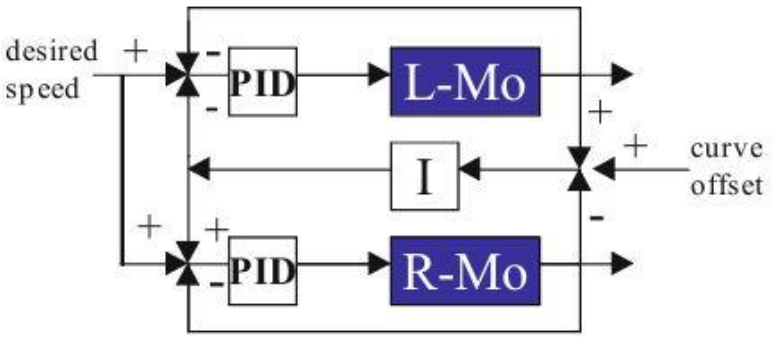


Figure 4-21: Motor Control Schema

Listing A-6 shows the implementation of Figure 4-21 in Arduino Mega compatible code.

4.2.2 Localisation

Localisation on the mobile platform is achieved by implementing the traditional dead reckoning method. Data from the wheel encoders are converted into linear distance travelled by each wheel. These travelled distances are then converted into Δx, Δy, Δϕ values which are added to the previous pose of the platform to determine the new pose (Listing A-7).

The new pose is serially transmitted to the Algorithm Layer where the dead reckoning data are fused with the sonar sensor data and particle filter data to achieve localisation [25].

4.2.3 Sensor Skirt

The sensor skirt implemented on this robot primarily consists of two types of sensors: 1) A Microsoft XBOX 360 Kinect V1 sensor (Figure 4-22) which will provide data for local navigation and local obstacle avoidance, and 2) an array of 7 ultrasonic sensors (US) which will be responsible for global localisation.

The XBOX 360 Kinect V1 sensor makes use of infrared light to create a depth map, while the ultrasound sensors use sound to determine distances to objects. Using these different detecting technologies ensures that the shortcoming of one sensor is mitigated by another sensor and therefore ensures that the robot does not break down catastrophically due to sensor malfunction, but that it carries on, albeit at a reduced performance rate. This technique is known as graceful degradation [39].

4.2.3.1 XBOX 360 Kinect V1 Sensor

The XBOX 360 Kinect V1 sensor is a depth camera where the depth detection capability consists of two parts [45]:

- An infrared laser, and
- An Infrared camera.

There is also an RGB camera which is essentially a normal web cam located close to the IR camera which allows normal RGB data to be captured.

Figure 4-22: XBOX 360 Kinect V1 Sensor

The XBOX 360 Kinect V1 sensor has a FOV of 57° in the horizontal axis and 43° on the vertical axis with a depth resolution of 640x480 pixels. Range values received from the XBOX 360 Kinect V1 sensors are in millimetres and are measured perpendicular to the major axis of the

sensor. The XBOX 360 Kinect V1 sensor dead spot is below 800 mm with a maximum detection distance of up to 4 m.

Data from the XBOX 360 Kinect V1 sensors are sent via USB to the attached PC to be used by the Algorithm Layer, updating the map with possible obstacles.

4.2.3.2 Ultrasonic Sensors (US)

The Ultrasonic Sensor array consists of 7 SRF02 ultrasonic sensors [54], attached to an adjustable upper deck. These sensors are connected to an Arduino Mega microcontroller via a serial port. Data from this array is sent via serial port to the Algorithm Layer and used to update the Particle Filter localiser.

Ultrasonic sensors are slow sensors due to the use of sound which travels at approximately 330 m/s. The maximum detection distance of the SRF02 sensors is approximately 2.5 m when detecting a pipe that has a 550 mm diameter. At this distance, it will take about 8 ms to detect an obstacle [54]. Implementing 7 ultrasonic sensors in a traditional way of detecting distance will take close to 60 ms. This can become a problem when the mobile platform needs to react quickly due to the speed at which it moves.

To speed up detection times, a scheme is used whereby sensors are requested to ping for obstacles but keep the distance data until it is requested. Furthermore, a staggered pattern is employed to minimise cross-talk. Each sensor has a unique address, as indicated in Figure 4-23, while distance data is requested using the following order: 3, 0, 4, 1, 5, 2, 6.

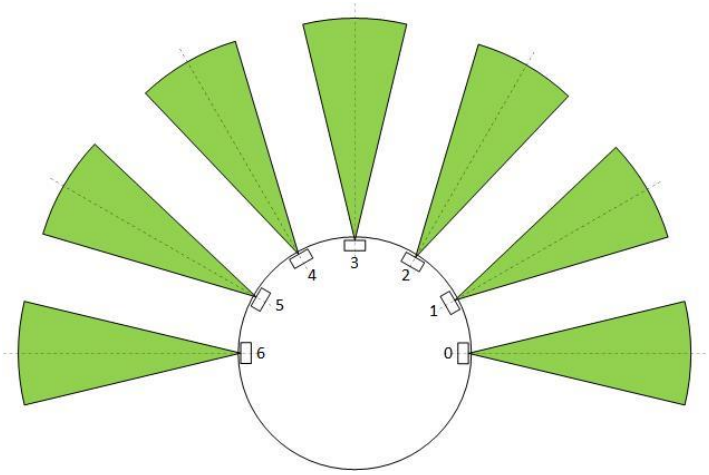


Figure 4-23: Sonar Sensor Firing Order

Listing A-8 shows the code implemented on an Arduino Mega 256 to interface to the SRF02 sensors using a serial bus. The received data is serially sent, using a different serial port to the PC for use by the Particle filter localiser.

4.3 Chapter Summary

This chapter summarises the design of the simulator and the algorithm layer, discussing the implementation of the researched techniques.

This design of an Algorithm Layer, as proposed by National Instruments, is discussed as part of the total control structure for the autonomous book truck. It highlights the design considerations and how these considerations were implemented. It also shows that it is possible to program an Algorithm Layer with relative ease using software packages that do not have complicated installation procedures.

Design considerations for the AMR are discussed with the focus on the physical components used for the platform. Furthermore, this section highlights the use of a sensor skirt and the sensors necessary for localisation and obstacle avoidance.

In the next chapter, the results conclusions and recommendations will be discussed.

Chapter 5 – Results, Conclusions and Recommendations

Chapter 5 is divided into two sections which will 1) Discuss the method and results obtained from the implementation of the simulator and the AMR, 2) Summarise the research process by presenting the findings of this study.

5.1 Results

This section explains the method used to compare the book return data with the simulation data. In order to determine the results, data from the simulator and from the AMR was compared with data from the Library Information Systems (LIS), showing the number of books returned per month for the year 2016.

Data from the Library Information Systems (LIS), for the year 2016, was obtained showing the number of books returned per month. This data is from the open collection at the Tshwane University of Technology Soshanguve South campus library and only reflect books from the open collection.

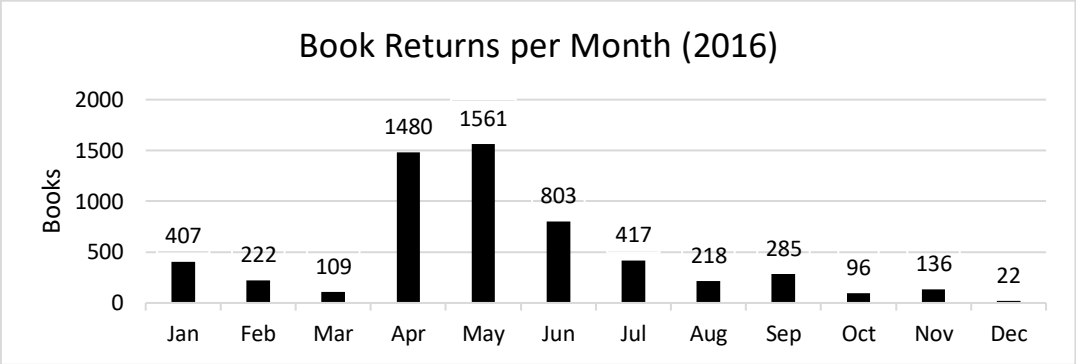
Initial data gathering for the AMR was done by simulating the AMR movement using the floor plan of the library at the Soshanguve South campus of Tshwane University of Technology (TUT). These movement times, generated during simulation, were recorded for comparison with the converted book return data.

Further data gathering, using the AMR, was done by implementing the Robotics Reference Architecture hardware.

5.1.1 Book Return Data

Table 5-1 shows the number of books returned per month for the open collection at the Soshanguve South library (SSoc).

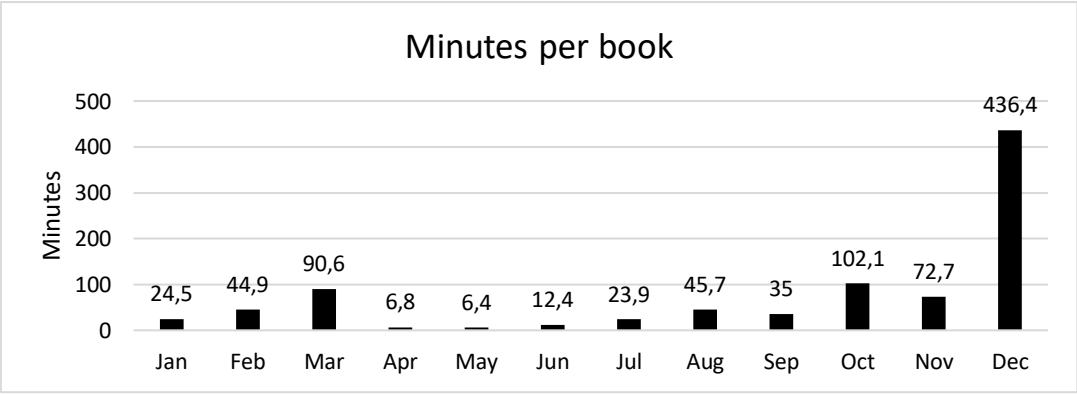
Table 5-1: Number of Books Returned per Month



In order to effectively compare the number of returned books, the data from Table 5-1 had to be converted into a time value. Table 5-2 shows the maximum allowed time that can be taken to shelve a book after it has been returned, in order to ensure the book will be available on the shelves the next working day.

The minutes shown in Table 5-2 are calculated using monthly book return from Table 5-1 and assuming a total of 21.8 working days per month with a typical working day of 8 hours. The minute values are an indication of how quickly a returned book should be shelved in order to have it available during the next working day. This data is per book and assumes that a shelver will, after each book returned, take that single book and shelve it, go back to the lending desk and then only pick up the next book to shelve. It does not take into account that a shelver might wait for multiple books before making the shelving trip. The reason for this was because the AMR only had a payload capacity of 1 book per trip and had to return to the lending desk to pick up the next book to drop off at the shelves.

Table 5-2: Maximum Shelving Time per Book



From Table 5-2 it can be seen that it should not take longer than 6.4 minutes per book to be returned to the shelves in order for them to be available the next working day.

5.1.2 Simulation Data

The simulation was done using the floor plan of the Soshanguve South campus library of the Tshwane University of Technology, as seen in Figure 5-1. The lending desk is on the ground floor while the open collection is on the first floor.

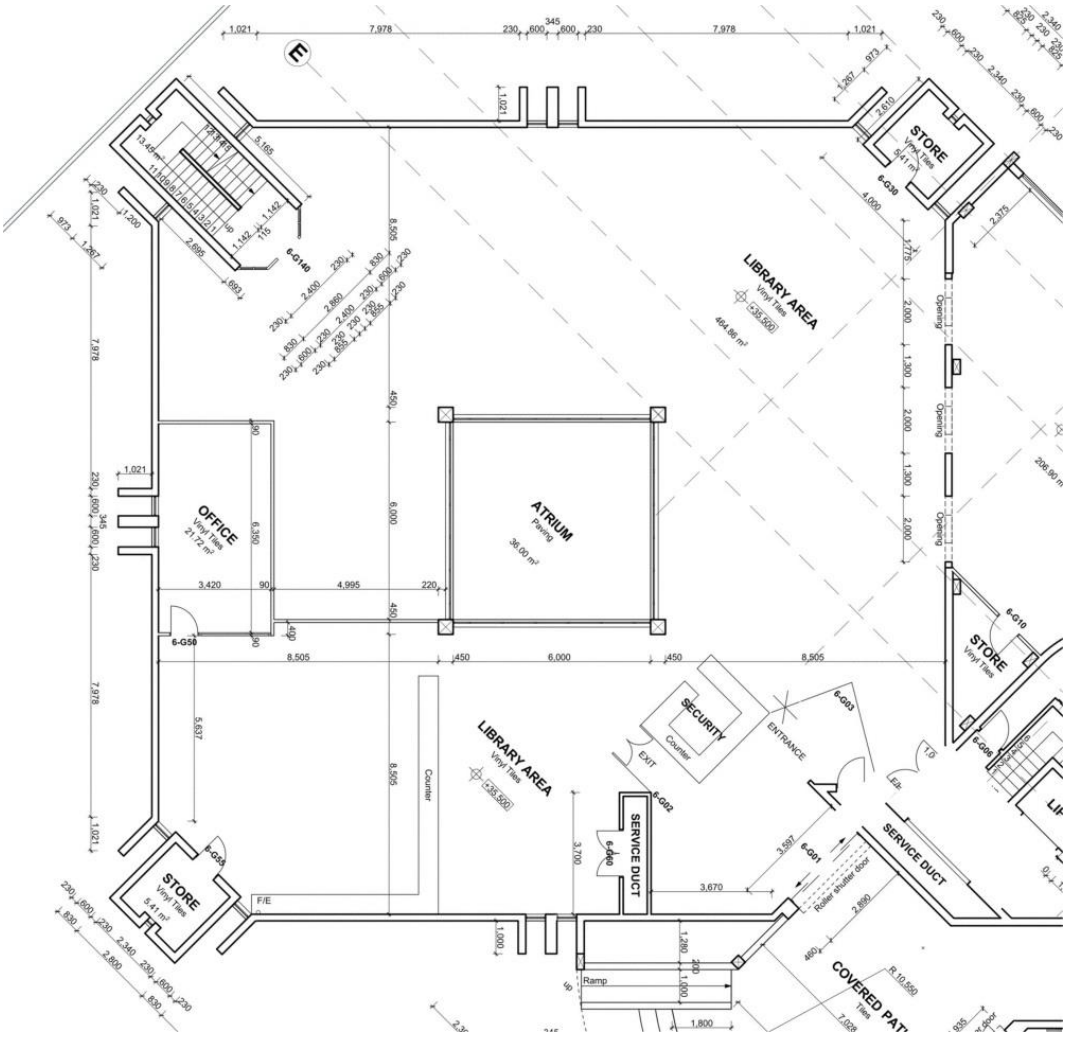


Figure 5-1: Library Floor Plan

The floor plan is created by scanning in an already existing paper-based architectural drawing, cleaning up the scanned image by removing measurement detail, annotations, etc. and then importing it into the simulator. After importing the cleaned up floor plan, the simulator

divided the floor plan into an occupancy grid to produce the tiles which were be used to show occupied space and the free spaces (Figure 5-2).

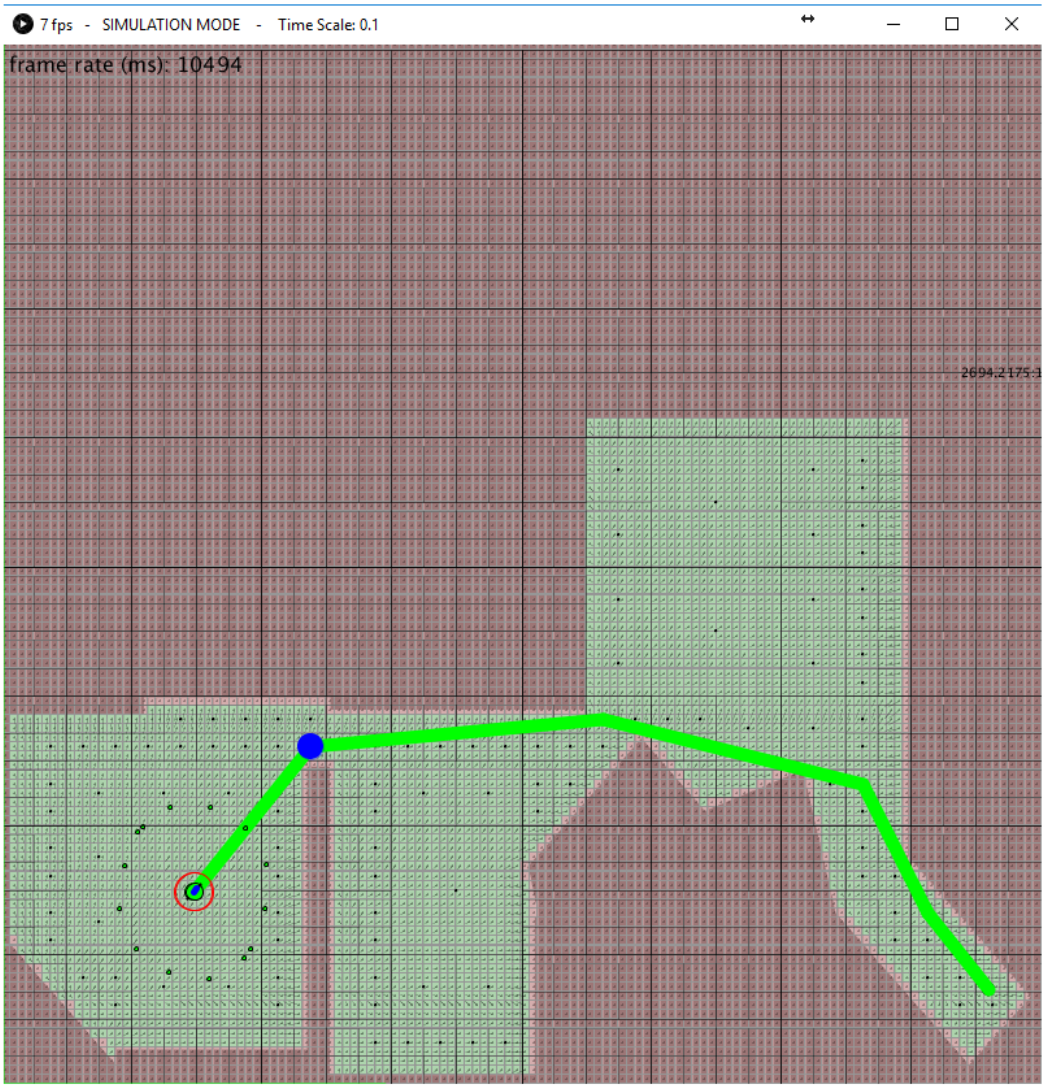


Figure 5-2: Converted Library Floor Plan (Ground floor)

To overcome the problem of the lending desk being on the ground floor while the open shelves are on the first floor, the simulation was divided into two parts. The first part simulated the robot moving to the stairs going up to the open shelves, and the second part simulated the movement from the stairs on the first floor to a drop-off point on the first floor. These two times were added together to produce a total one-way time which was then multiplied by two to produce a round-trip time back to the lending desk.

One of the delimitations of this study is that the mobile platform will not negotiate stairs in multilevel libraries. As a result, possible times used to negotiate stairs by the mobile platform were omitted when calculating the total time. The times were calculated assuming that both the ground and first floors of the library are on the same level.

In the simulation, the maximum movement speed of the robot was set to only 1 m/s, roughly the same as someone pushing a trolley full of books, while the starting point was randomly picked within a 1 m x 2 m area behind the lending desk, simulating the inexact starting point of the robot when returned from the drop-off point.

An average time was calculated using the total times of the ten simulation runs. These results can be seen in Table 5-3, clearly showing that the AMR could deliver a single book to the drop-off point within the time limit set to have the book available by the next working day.

Table 5-3: Results of Simulated Data

Lending Desk to Stairs		***Stairs to drop off bin***		***Total Return Data***	
Distance (cm)	***Time (s)***	***Distance (cm)***	***Time (s)***	***Distance (cm)***	***Time (s)***
2546.433	25.7	1867.719	19	8828.304	89.4
2544.989	25.6	1864.812	18.8	8819.602	88.8
2479.516	24.9	1868.632	19.1	8696.296	88
2464.285	24.8	1864.812	18.8	8658.194	87.2
2487.262	25	1866.515	18.9	8707.554	87.8
2424.416	24.4	1866.515	18.9	8581.862	86.6
2493.986	25.1	1864.812	18.8	8717.596	87.8
2415.776	24.4	1864.812	18.8	8561.176	86.4
2487.811	25.1	1867.719	19	8711.06	88.2
2534.231	25.5	1866.515	18.9	8801.492	88.8
			Averages	**8708.3136**	**87.9**

5.1.3 AMR Data Gathering

The following subsystems were tested using the AMR: 1) Obstacle detection and map update, and 2) Path planning to the drop off point.

These subsystems were tested independently to ensure that each subsystem functions correctly before being combined to form a complete functioning unit.

5.1.3.1 Obstacle detection and map update

A screenshot showing the video feed (left) and the RGB depth data (right) from the XBOX 360 Kinect V1 sensor can be seen in Figure 5-3. When comparing the depth data with the map being created in Figure 5-4, it is seen that the map is created according to the distance data received from the XBOX 360 Kinect V1 sensor.

The obstacles detected, in this case the shelves in the library, are represented by light-red grid squares. Each of these squares has a number printed inside the square which corresponds to the number of data points associated with that specific square. The bigger the number, the more confident the system is that there is an actual obstacle and that it is not just random sensor noise.

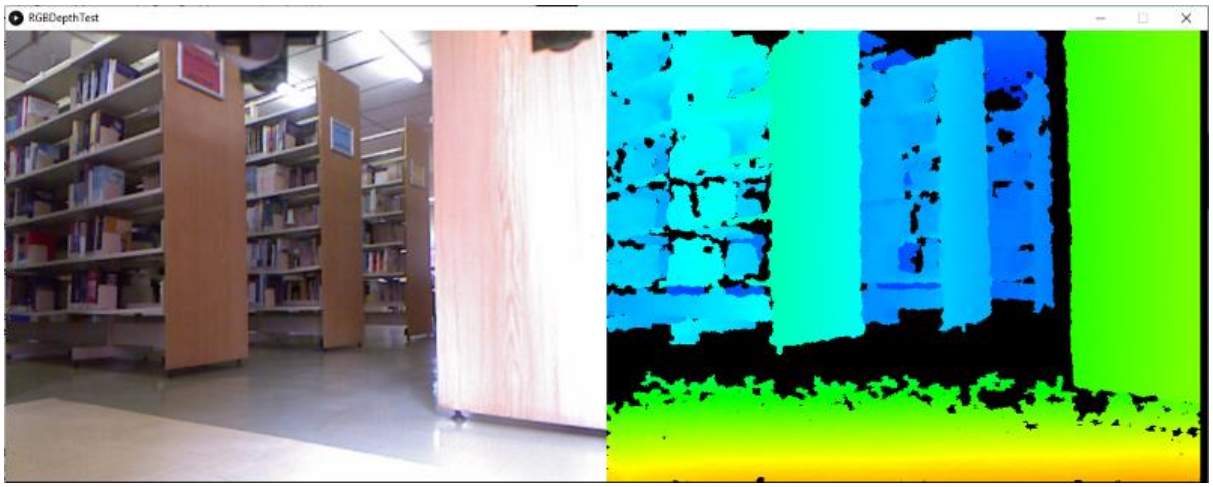


Figure 5-3: RGB Depth Test

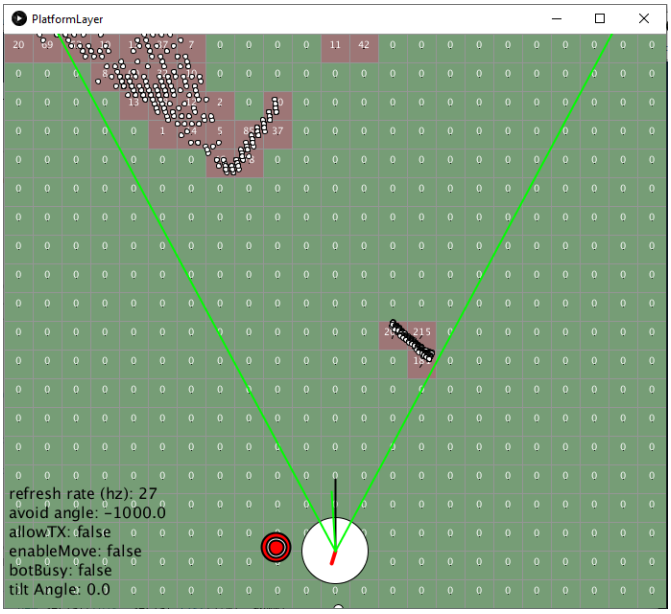


Figure 5-4: Map from depth data

5.1.3.2 Path planning to the drop-off point

The following images show the AMR in between shelves, correctly mapping its location.

Figure 5-5: AMR shown from opposite ends of the shelves

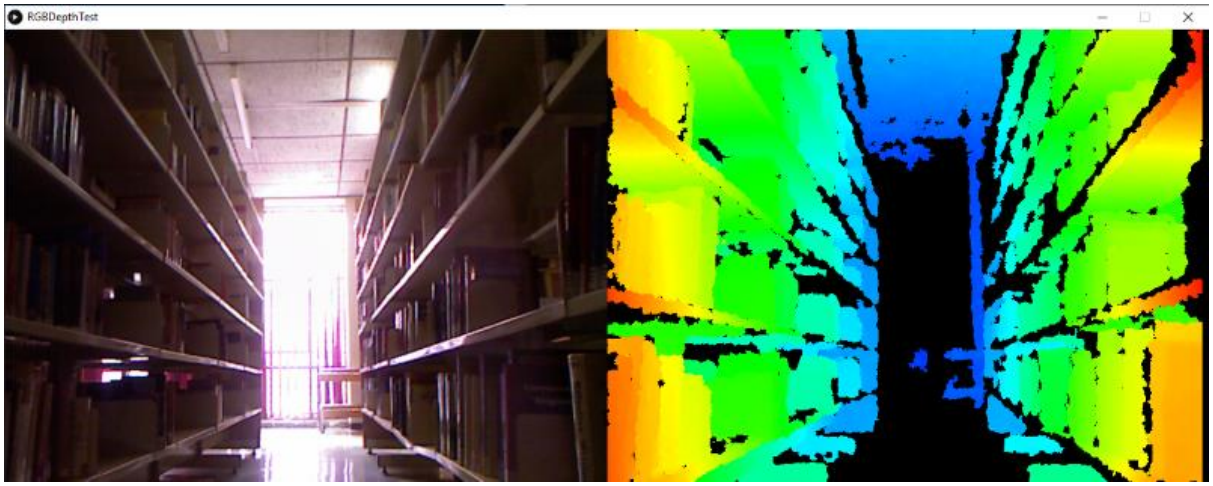


Figure 5-6: RGB depth data when in isle between shelves

From Figure 5-7, it is clear that the data received from the XBOX 360 Kinect V1 sensor is correctly mapped to the converted floor plan of the library. It also shows the path to the goal and the nodes created using the visibility graph node creation method.

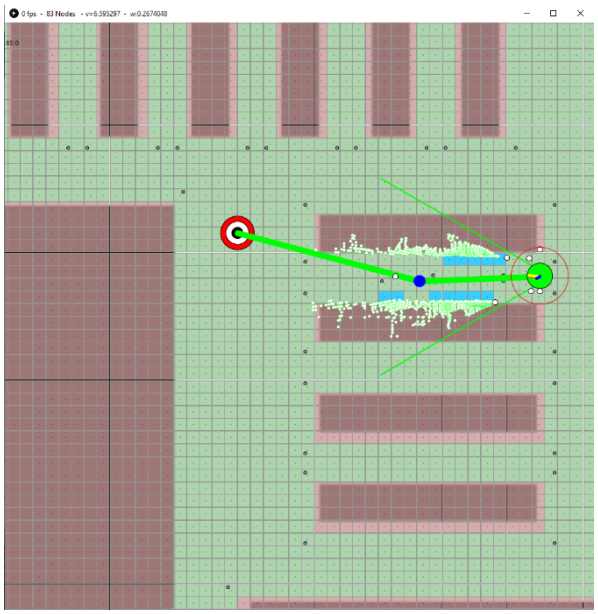


Figure 5-7: Shelf map update

5.2 Summary and Findings of the Research Process

The following sections will expand on the content from the previous sections by presenting a summary of the research process and by presenting the findings of this study.

5.2.1 Summary of the Research Process

As stated in Chapter 3, the Design and Creation research strategy was used in this research report.

The Design and Creation research method makes use of an iterative process consisting of 1) Becoming familiar with the research questions through a literature study, 2) Identifying possible solutions for each of the research questions, 3) Implementing the proposed solution for each research question, 4) Examining the implemented solution to determine its worth, and 5) Consolidating the evaluated results to select the best solution for each of the research questions.

5.2.2 Summary of the Findings of the Research Study

The findings of this research study are driven by the research objectives and the research questions. In this section, each of the findings will be discussed based on the research objectives.

Research Objective 1 - *Investigate current techniques used by mobile robots to navigate and move, and then identify the most appropriate techniques for a library environment.*

In Chapter 2, an in-depth overview of the literature that supports this research is supplied. It was shown that library books take between 1 and 5 days, after being returned, to get shelved for re-issue. Because the intention of this research was to decrease library book dead time using an automated book truck, research was conducted into the current and historic use of robotics in libraries. It was found that the current research focuses on 1) Inventory control, 2) Book manipulation, and 3) Library assistants. None of the current research focuses on moving books back from the lending desk to the shelves.

Investigating AMR movement techniques currently being used in libraries highlighted the use of wheeled motion, specifically using the differential drive configuration. It was decided to follow the current trend in library AMRs and also implement wheeled motion.

Research Objective 2 - *Create a simulated environment of an operational library. This simulation will include a model of the robot incorporating the identified techniques to simulate the validity of the selected techniques.*

A robotic simulator, capable of simulating the designed AMR, was created from the ground up. It incorporates a model of the robot which includes the modelling of the ultrasonic sensors, the XBOX 360 Kinect V1 sensor and the drive mechanics. Furthermore, it consists of subsystems which convert a JPG image into an occupancy grid map, localise the robot in the map using the ultrasonic distance data, plan a path between the destination and the goal using the A-star path planning method, and update the map based on data from the XBOX 360 Kinect V1 sensor.

Simulations of the AMR in the simulated library environment show that books can be returned fast enough to enable faster library book turn-around times.

Research Objective 3 - *Apply the selected techniques on a robotic platform to verify the simulated results*

The different subsystems were implemented and tested on the AMR. Due to extreme latency experienced between the different layers and subsystems, it was not possible to combine all these subsystems into an effective application using the selected hardware components.

Since the software libraries concerned with aspects such as localisation and movement had to be developed for this study, these developed libraries were not effectively optimised or as efficient as commercially available software libraries or software libraries found in the already existing robotic software. Using existing software libraries might have solved the system latency problem.

Research Objective 4 - *Draw conclusions on the effectiveness of an AMR in the library environment.*

This chapter fulfils this objective.

5.3 Chapter Summary

This chapter summarises the different data sets used to create the results. A summary of the research process and research findings is also provided. The research methodology used to guide this research is described and summarised.

Chapter 6 – Conclusions Based on the Research Process and the Research Findings

This section provides the conclusions based on the findings. Conclusions based on the research process are discussed first, followed by the conclusions based on the research questions.

6.1 Conclusions Related to the Research Process

The conclusions relating to the quantitative research process are as follows:

The research approach: Since the research focused on reducing the dead time of library books and not on user experience, a quantitative approach was the correct method to use.

The research question: The researcher concludes that the research question and sub-questions were clear because it provided guidance and focus to complete the study.

The research goal and objectives: Using the research question, a research goal and research objectives were formulated which were used to determine the answers to the research question and sub-questions. It was concluded that the goal and objectives were clear and sufficient.

The method of data collection: In this research effort, data was collected using a simulation of an AMR in a library environment. The analysed simulation data showed that the dead time of books can be reduced by a significant factor.

It should be noted that the data was only collected using the simulated floor plan of one library and although the data is favourable towards the implementation of an AMR in this library, it may differ in other libraries. Further studies should include the use of other library floor plans.

The process of data analysis: The Systems Development Method from the Design and Creation research approach was followed to decide on the components used and the effectiveness of each subsystem.

The process of calculating the time necessary to shelve a book and compare that with the time the AMR will take to shelve a book provided the conclusion that an AMR will reduce library book dead time.

6.2 Conclusions Related to the Findings of the Research Study

The research question leads to the research objectives which in turn leads to findings and, ultimately, to conclusions which answer the research questions. In this section, the conclusion of each research question is discussed.

- **Research sub-question 1:** What will the robotic platform look like?

  From the findings in section 2.3.3, Platform and Sensors, it is concluded that the differential drive configuration should be used for the robotic platform. The differential drive is stable and power efficient and since the floor of the library is flat, this configuration will not have any driving issues. The platform should also have roughly the same size footprint as a human, which will allow for a large enough payload area to carry books.

- **Research sub-question 2:** Which sensors will be used to facilitate localisation and obstacle avoidance?

  From the findings in section 2.3.3, Platform and Sensors, it was decided to use active ranging techniques instead of cameras, in order to eliminate the problem of inconsistent lighting. Light shining into windows can create bright patches on the floors and against shelves, making the camera ineffective due to camera image contrast limitations.

  By making use of the XBOX 360 Kinect V1 sensors and sonar sensors, two different technologies are used to localise and detect obstacles. This will ensure effective redundancy and graceful degradation.

- **Research sub-question 3:** Which mapping and localisation techniques or combination of techniques are the best to implement?

  From the findings in the section 2.3.1, Mapping and Localisation Techniques, it is concluded that particle filters are superior to the use of other localisation techniques due to its ease of implementation, robust nature, its ability to globally localise, and that it does not limit the location hypothesis to only one position.

  From the findings in section 2.3.2, Path Planning, it is concluded that the A-star path finding method should be used. The distance graph is created using nodes as identified by the visibility graph method since this method creates the least number of nodes, increasing the speed of the whole path finding algorithm. Potential fields will be used for local path planning since the map can be augmented with additional data, as provided by the sensors on the robot and since an a priori map is already available.

- **Research sub-question 4:** Can the robot be implemented in a typical library setup without library remodelling?

  Although large scale library remodelling will not be necessary, measurements of various doorways and the distances between shelves indicated that some aspects of a library might need some adjustment in order to have the AMR move without collision.

  Unfortunately, due to the extreme subsystem latency, this could not be tested and confirmed using the AMR.

6.3 Recommendations

In this section, recommendations based on the conclusions from the previous section are presented based on 1) The research process, 2) Research findings, and 3) Future research.

6.3.1 Recommendations Relating to the Research Process

In this research study, the Design and Creation research method was implemented as the research method. The researcher recommends this method for any research study that involves the design, testing and implementation of an intervention.

6.3.2 Recommendations Relating to the Research Findings

The simulator was designed and an AMR was simulated which included sensors, drive kinematics, and a real world library floor plan. The researcher suggests that additional simulations should be done using floor plans from a number of different real world libraries.

6.3.3 Recommendations Relating to Future Research

Due to “Processing” being designed for quick graphical sketches, the implementation for the AMR should be moved to robotic specific software to verify the results.

The initial reason for deciding to use “Processing” was the creation of a simple simulator to be used without elaborate installation processes and which could be used to quickly test simple robotic applications. However, using one of the many available robotic packages would have achieved the same outcome but probably with better optimised results and probably in a shorter amount of time.

The proposed and built system can be used as the base for all the aforementioned research by providing an autonomous robot capable of retrieving and replacing books, identifying books in the wrong spots, and detecting books left on tables and study cubicles by implementing technologies like RFID.

Since this is a mobile platform capable of navigating an entire library, it can be used to notify library staff if intervention in book handling is needed, or to integrate discovered books left

on library study desks, which in turn can show patrons and library staff where certain sought-after books are. This map can then be integrated with the LIS, making the task of finding books easier.

Furthermore, the AMR can be enhanced by including a ‘pickup’ function whereby users at desks can request a pick-up from the AMR. These picked up books can then be moved to the collection bins for shelving at a later stage.

Data gathered can be used to determine user patterns and optimise library layout.

Although not implemented, bumper sensors and stasis sensors should be implemented to safeguard against collisions which might be missed by the major sensors.

6.4 Concluding Remarks

In this research effort, it was proven that an AMR can reduce library book dead time when implemented to move books from the lending desk back to drop off points in the shelves.

6.5 Chapter Summary

The findings made during this research effort are described and conclusions are made which provides answers to the research questions asked in Chapter 1. Future study recommendations, which were based on the findings and conclusions, were made to develop this research effort further.

References

[1] C. Cook and F. M. Heath, “Users’ perceptions of library service quality: A LibQUAL+ qualitative study,” *Library Trends*, vol. 49, no. 4, p. 548, 2001.

[2] L. S. White, “Report on Shelving Project,” 1999. [Online]. Available: http://www.lib.virginia.edu/mis/benchmarking/bench-shortrpt.html. [Accessed: 07-Aug-2015]

[3] J. Ho and G. H. Crowley, “User perceptions of the ‘reliability’ of library services at Texas A&M University: A focus group study,” *The journal of academic librarianship*, vol. 29, no. 2, pp. 82–87, 2003.

[4] F. N. Onifade, G. O. Onifade, and B. O. Akintola, “Staff Attitude to Shelving and Shelf Reading in Academic Libraries,” *North Carolina Libraries*, vol. 68, no. 2, p. 12, 2010.

[5] H. Moradi, K. Kawamura, E. Prassler, G. Muscato, P. Fiorini, T. Sato, and R. Rusu, “Service robotics (the rise and bloom of service robots),” *Robotics & Automation Magazine*, vol. 20, no. 3, pp. 22–24, 2013.

[6] M. Arumugaraja, B. GuguPriya, and M. Soundarya, “The Library Management Robot,” *International Journal of Engineering and Computer Science*, vol. 3, no. 3, pp. 5008–5012, 2014.

[7] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[8] C. A. Cuadra, D. Black, H. Borko, A. W. Luke, J. Mersel, F. Neeland, D. R. Pascale, A. Teplitz, R. F. Von Buelow, and E. M. Wallace, “Technology and Libraries.,” 1967.

[9] M. K. Sinha and A. Chanda, “Exploring RFID Technology Application for Managing Library and Information Services in University and Institutional Libraries of North East India: An Overview,” *Sinha, Manoj Kumar and Chanda, Anupam (2014). Exploring RFID Technology Application for Managing Library and Information Services in University and Institutional Libraries of North East India: An overview. International Journal of Information Sources and Services*, vol. 1, no. 3, 2014.

[10] E. Iglesias, *Robots in Academic Libraries: Advancements in Library Automation*. IGI Global, 2013.

[11] R. Hansson, “Robot lends a hand in a Swedish library,” *Industrial Robot: An International Journal*, vol. 22, no. 5, pp. 34–35, 1995.

[12] J. Suthakorn, S. Lee, Y. Zhou, R. Thomas, S. Choudhury, and G. S. Chirikjian, “A robotic library system for an off-site shelving facility,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, 2002, vol. 4, pp. 3589–3594.

[13] T. Tomizawa, A. Ohya, and S. Yuta, “Book browsing system using an autonomous mobile robot teleoperated via the internet,” in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 2002, vol. 2, pp. 1284–1289.

[14] U. Nehmzow, *Mobile robotics: a practical introduction*. Springer Science \& Business Media, 2012.

[15] R. Ramos-Garijo, M. Prats, P. J. Sanz, and A. P. Del Pobil, “An autonomous assistant robot for book manipulation in a library,” in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003, vol. 4, pp. 3912–3917.

[16] K. H. Yuan, A. C. Hong, M. Ang, and G. S. Peng, “Unmanned library: an intelligent robotic books retrieval \& return system utilizing RFID tags,” in *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, 2002, vol. 4, p. 5–pp.

[17] J. Behan and D. T. O’Keeffe, “The development of an autonomous service robot. Implementation:‘Lucas’—The library assistant robot,” *Intelligent Service Robotics*, vol. 1, no. 1, pp. 73–89, 2008.

[18] M. Prats, E. Martinez, P. J. Sanz, and A. P. Del Pobil, “The UJI librarian robot,” *Intelligent Service Robotics*, vol. 1, no. 4, pp. 321–335, 2008.

[19] W. Lin, H.-P. Yueh, H.-Y. Wu, and L.-C. Fu, “Developing a service robot for a children’s library: A design-based research approach,” *Journal of the Association for Information Science and Technology*, vol. 65, no. 2, pp. 290–301, 2014.

[20] M. Debczak, “New York Library Installs Tiny Trains to Deliver Books,” 2016. [Online]. Available: http://mentalfloss.com/article/86601/new-york-public-library-installs-tiny-

trains-deliver-books. [Accessed: 28-May-2018]

[21] C. Low, “The New York Public Library has a mini roller coaster for books,” 2016. [Online]. Available: https://www.engadget.com/2016/10/01/new-york-public-library-book-train/. [Accessed: 28-May-2018]

[22] M. Sreejith, S. Joy, A. Pal, B.-S. Ryuh, and V. S. Kumar, “Conceptual Design of a Wi-Fi and GPS Based Robotic Library Using an Intelligent System,” *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 12, pp. 2338–2342, 2015.

[23] R. Li, Z. Huang, E. Kurniawan, and C. K. Ho, “AuRoSS: an Autonomous Robotic Shelf Scanning System,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015, pp. 6100–6105.

[24] E. Blakemore, “High Tech Shelf Help: Singapore’s Library Robot,” 2016. [Online]. Available: https://lj.libraryjournal.com/2016/08/industry-news/high-tech-shelf-help-singapores-library-robot/. [Accessed: 23-May-2018]

[25] T. Bräunl, *Embedded robotics: Mobile robot design and applications with embedded systems*. Springer Science & Business Media, 2008.

[26] N. Sariff and N. Buniyamin, “An overview of autonomous mobile robot path planning algorithms,” in *Research and Development, 2006. SCOReD 2006. 4th Student Conference on*, 2006, pp. 183–188.

[27] J. J. Leonard and H. F. Durrant-Whyte, “Mobile robot localization by tracking geometric beacons,” *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[28] J. Borenstein, H. Everett, and L. Feng, “Where am I? Sensors and methods for mobile robot positioning,” *University of Michigan*, vol. 119, p. 120, 1996.

[29] A. Burguera, Y. González, and G. Oliver, “Mobile Robot Localization using Particle Filters and Sonar Sensors,” *Advances in Sonar Technology*, pp. 213–232, 2009.

[30] J.-S. Gutmann and D. Fox, “An experimental comparison of localization methods continued,” in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference*

on, 2002, vol. 1, pp. 454–459.

[31] S. Thrun, “Particle filters in robotics,” in *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002, vol. 1.

[32] W. Greg and B. Gary, “An introduction to the Kalman filter,” *Department of Computer Science, University of North Carolina at Chapel Hill, NC*, 2006.

[33] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 1999, vol. 2, pp. 1322–1328.

[34] A. Bansail, “Monte Carlo localization for mobile robots in dynamic environments,” Texas Tech University, 2002.

[35] I. M. Rekleitis, “A particle filter tutorial for mobile robot localization,” *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.

[36] M. I. Ribeiro, “Obstacle avoidance,” *Instituto de Sistemas e Robótica, Instituto Superio Técnico*, p. 1, 2005.

[37] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[38] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[39] J. L. Jones, *Robot Programming: A Practical Guide to Behaviour-Based Robotics*. TAB Robotics, 2004.

[40] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[41] T. W. Gussu and C.-Y. Lin, “Geometry Based Approach to Obstacle Avoidance of Triomnidirectional Wheeled Mobile Robotic Platform,” *Journal of Sensors*, vol. 2017, 2017.

[42] M. Pospisilik, P. Varacha, P. Bartonik, J. Vorisek, and P. Neumann, “Testing of a Reliability of SRF02 Ultrasonic Detectors,” *International Journal of Systems Applications, Engineering & Development*, vol. 7, no. 4, pp. 175–182, 2013.

[43] J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[44] “Photonic Frontiers: Gesture Recognition.” [Online]. Available: https://www.laserfocusworld.com/articles/2011/01/lasers-bring-gesture-recognition-to-the-home.html. [Accessed: 04-Sep-2018]

[45] T. Taylor, *Robotics Developer Studio Reference Platform Design V1.0 RDS 4*. 2012 [Online]. Available: http://go.microsoft.com/fwlink/?LinkID=228540&clcid=0x409

[46] R. A. El-laithy, J. Huang, and M. Yeh, “Study on the use of Microsoft Kinect for robotics applications,” in *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, 2012, pp. 1280–1288.

[47] National Instruments, “A Layered Approach to Designing Robot Software,” 2012. [Online]. Available: http://www.ni.com/white-paper/13929/en/. [Accessed: 20-May-2015]

[48] B. J. Oates, *Researching Information Systems and Computing*. SAGE Publications, 2006.

[49] J. Mouton, *How to succeed in your Master’s & Doctoral Studies*, 1st ed. Van Schaik, 2001.

[50] Parallax Inc., *Robot Base Kit*, v1.0 ed. 2011 [Online]. Available: https://www.parallax.com/product/28977. [Accessed: 02-Sep-2012]

[51] Parallax Inc., *Motor Mount and Wheel Kit with Position Controller*, v1.3 ed. 2009 [Online]. Available: http://asrob.uc3m.es/images/d/dd/Motor_paralax.pdf. [Accessed: 02-Sep-2012]

[52] Parallax Inc., *37.5-Position Quadrature Encoder Set (#29321)*, v2.0 ed. 2016 [Online]. Available: https://www.parallax.com/sites/default/files/downloads/29321-36-Pos-Encoder-Set-v2.0.pdf. [Accessed: 15-Mar-2017]

[53] M. Kellet, “Interfacing Microcontrollers with Incremental Shaft Encoders,” 2001. [Online]. Available: http://mkesc.co.uk/ise.pdf. [Accessed: 15-Aug-2012]

[54] “SRF02 Ultrasonic Range Finder.” [Online]. Available: http://robot-electronics.co.uk/htm/srf02tech.htm. [Accessed: 28-Jun-2017]

Appendix A

Appendix A contains code segments for reference purposes and to explain and clarify flow charts.

Listing A-1: Occupancy grid map create program segment

```
1.  //###Generate Grid Map
2.  img = loadImage(mapName);          //Loads image
3.
4.  //### Resize image to image width and height represented by the world
5.  img.resize(int(imgWidth), int(imgHeight));
6.
7.  surface.setResizable(true);
8.  surface.setSize(int(graphicBoxWidth), int(graphicBoxHeight));
9.
10. //##Calculates the number of tiles in X and in Y
11. maxTilesX = ceil((float(img.width)/(tileSize)));
12. maxTilesY = ceil((float(img.height)/(tileSize)));
13.
14. //## Calculates the smallest binary value that can be used to calculate the tilesize
     width
15. int xx = 0;
16. while (int(pow(2,xx)) < int(maxTilesX))
17. {
18.   xx++;
19. }
20.
21. int yy = 0;
22. while (int(pow(2,yy)) < int(maxTilesY))
23. {
24.   yy++;
25. }
26. tileSize = img.width/pow(2,xx);
27.
28. //##Calculates the number of tiles in X and in Y
29. maxTilesX = ceil((float(img.width)/(tileSize)));
30. maxTilesY = ceil((float(img.height)/(tileSize)));
31.
32. println("img.Width : "+img.width+", img.Height: "+img.height);
33. println("scaleFactor :"+scaleFactor);
34. println(maxTilesX+","+maxTilesY);
35.
36. tile = new Tile[maxTilesX][maxTilesY];
37.
38. //## Set the starting position of the tiles so tiles will start in the bottom left c
    orner of the map
39. float _startX = tileSize/2;
40. float _startY = tileSize/2;
41. println("startX: "+_startX+", startY: "+_startY);
42.
43. delay(2000);
44.
45. //###Sets up a 2D array which will hold the world Tiles
46. for (int x = 0; x < maxTilesX; x++)
47. {
48.   for (int y = 0; y < maxTilesY; y++)
49.   {
50.     tile[x][y] = new Tile((_startX + tileSize * x), (_startY +  y * tileSize));
51.   }
```

```
52. }
53.
54. //Scans the pixels of the background image to build the occupancy grid
55. img.filter(THRESHOLD, 0.9);              //Convert image to greyscale
56. for (int x = 0; x < imgWidth; x++)
57. {
58.   for (int y = 0; y < imgHeight; y++)
59.   {
60.     color c = img.get(x,y);
61.     if (c == color(0))
62.     {
63.       int tileX = floor((x) / tileSize);
64.       int tileY = floor((imgHeight - 1 - y) / tileSize);
65.       if ((tileX >= 0) && (tileY >= 0))
66.       {
67.         tile[tileX][tileY].gravity = 1;
68.         tile[tileX][tileY].tileType = "MAP";      //Set tileType to PERMANENT/MAP OB
    STACLE
69.         tile[tileX][tileY].update();
70.       }
71.     }
72.   }
73. }
```

Listing A-2: Particle filter suitability calculation

```
1.  //###Calcualtes distances to obstacles for each sensor in the sensor array
2.  //###  This function is used by the simulated              robot               and
    particles to sense distance to obstacles
3.  void sense()
4.  {
5.    for (int k = 0; k < sensors.size(); k++)
6.    {
7.      sensors.get(k).sense(location.x,location.y,heading);
8.      if ((sensors.get(k).sensorObstacleDist <= safeDistance) && (nodeType == "ROBOT")
    ) myRobot.collisionFlag = true;
9.    }
10. }
11. //Calculates the probability of how closely a particle's measurements to an obstacle
     coresponds with that of the robot.
12. //A prob value is calculated for each sensors distance which is multiplied to all ot
    her probabilities of the specific particle
13. //Uses a gausian with:
14. //  mu      - Particle's measured distance to a obstacle
15. //  sigma   - Particle's measurement noise
16. //  x       - Robot's distance measurement of the same sensor
17. void measureProb()
18. {
19.   prob = 1.0;         //Set probability to maximum value
20.   for (int k = 0; k < sensors.size(); k++)
21.   {
22.     float mu = sensors.get(k).sensorObstacleDist;
23.     float sigma = sensors.get(k).sensorNoise;
24.     float x = myRobot.sensors.get(k).sensorObstacleDist;
25.     float tempProb = exp(- (pow(mu - x, 2) / pow(sigma,2)/2.0) / sqrt(2*PI * pow(sig
    ma,2)));
26.     prob *= tempProb;
27.   }
28. }
```

Listing A-3: doQuadTree function

```
1.  int QuadTreeLevel = 5;
2.  ArrayList<Integer> closedList = new ArrayList<Integer>();
3.  ArrayList<Node> allNodes = new ArrayList<Node>();
4.  ArrayList<Integer> openList = new ArrayList<Integer>();
5.  ArrayList<Integer> finalPath = new ArrayList<Integer>();
6.
7.  //Recursively generates the quad tree nodes
8.  void doQuadTree(int _btmLeftX, int _btmLeftY, int _sizeW, int _sizeH, int _level)
9.  {
10.   //If no mix is found in a quad - draw a node
11.   if (!findMix(_btmLeftX, _btmLeftY, _sizeW, _sizeH))
12.   {
13.    float nodeX = (_btmLeftX + float(_sizeW)/2)*tileSize;  //cast _sizeW as float in
    order to do math
14.    float nodeY = (_btmLeftY + float(_sizeH)/2)*tileSize;  //cast _sizeH as float in
    order to do math
15.
16.    allNodes.add(new Node(nodeX,nodeY,allNodes.size()));    //Add new node to allNode
    s arrayList
17.    return;
18.   }
19.   //If a mixed quad is found and it is the last level DO NOT draw a node, just retur
    n
20.   else if (findMix(_btmLeftX, _btmLeftY, _sizeW, _sizeH) && _level == 0)
21.   {
22.     return;
23.   }
24.   else
25.   {
26.     //If a mixed quad is found: Divide the quad into four new quads
27.     //Btm left quad
28.     doQuadTree(_btmLeftX, _btmLeftY, _sizeW/2, _sizeH/2, _level-1);
29.
30.     //Btm right quad
31.     doQuadTree(_btmLeftX + _sizeW/2, _btmLeftY, _sizeW/2, _sizeH/2, _level-1);
32.
33.     //Top left quad
34.     doQuadTree(_btmLeftX, _btmLeftY + _sizeH/2, _sizeW/2, _sizeH/2, _level-1);
35.
36.     //Top right quad
37.     doQuadTree(_btmLeftX + _sizeW/2, _btmLeftY + _sizeH/2, _sizeW/2, _sizeH/2, _leve
    l-1);
38.   }
39. }
40.
41. //Function returns TRUE if an occupied tile is found within a certain square of tile
    s
42. boolean findMix(int _btmLeftX, int _btmLeftY, int _sizeW, int _sizeH)
43. {
44.   noFill();
45.   stroke (0);
46.   strokeWeight(0);
47.
48.   //###Draws the quad tree divisions on the screen
49.   rectMode(CORNERS);
50.   rect (toScreenX(int(_btmLeftX * tileSize)), toScreenY(int(_btmLeftY * tileSize)),

51.         toScreenX(int(_btmLeftX * tileSize + _sizeW * tileSize)), toScreenY(int(_btm
    LeftY * tileSize + _sizeH * tileSize)));
52.
```

```
53.
54.   for (int x = 0; x < _sizeW; x++)
55.   {
56.     for (int y = 0; y < _sizeH; y++)
57.     {
58.       if (tile[x+_btmLeftX][y+_btmLeftY].tileType == "MAP" || tile[x+_btmLeftX][y+_b
    tmLeftY].tileType == "USER")
59.       {
60.         return true;
61.       }
62.     }
63.   }
64.   return false;
65. }
```

Listing A-4: Movement vector calculation

```
1.  //## Calculates the movement vector based on the robot position and repulsive and at
    tractive forces
2.  vectorAOFWD.x = (calcAttractField(myRobot.location.x, myRobot.location.y).x + calcRe
    pulsiveField(myRobot.location.x, myRobot.location.y).x);
3.  vectorAOFWD.y = (calcAttractField(myRobot.location.x, myRobot.location.y).y + calcRe
    pulsiveField(myRobot.location.x, myRobot.location.y).y);
4.
5.  // Based on Games Programming: Methods and How to's - Dr James Jordaan Revision 4.1
    p196
6.  angleToGoal = atan2(vectorAOFWD.y,vectorAOFWD.x) - myRobot.heading;
7.  if (angleToGoal < (-PI)) angleToGoal += 2*PI;
8.  if (angleToGoal > (PI)) angleToGoal -= 2*PI;
9.
10. //###Calculates the magnitude of the AOFWD vector to determine speed
11. velocityToGoal = vectorAOFWD.mag();
```

Listing A-5: Shaft encoder implementation

```
1.  //Array used to increment encoder values
2.  const int encoderArray[]={0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
3.
4.  void setup()
5.  {
6.    setupMotorControl();
7.    leftWheelServo.writeMicroseconds(1500);
8.    rightWheelServo.writeMicroseconds(1500);
9.    //### Comm port for comms to PC
10.   Serial.begin(115200);
11.   Serial.println("PC Comms Started:");
12. }
13.
14. void setupMotorControl()
15. {
16.   //Attaches the encoder inputs to interrupts in order to catch state changes
17.   //Ints attached to pins 2,3 and 20,21
18.   //If int fires, ISR readEncoder will be called and executed
```

```
  attachInterrupt(0,readencoder, CHANGE);
  attachInterrupt(1,readencoder, CHANGE);
  attachInterrupt(2,readencoder, CHANGE);
  attachInterrupt(3,readencoder, CHANGE);

  pinMode(chn_l_a,INPUT);
  digitalWrite(chn_l_a,HIGH);
  pinMode(chn_l_b,INPUT);
  digitalWrite(chn_l_b,HIGH);
  pinMode(chn_r_a,INPUT);
  digitalWrite(chn_r_a,HIGH);
  pinMode(chn_r_b,INPUT);
  digitalWrite(chn_r_b,HIGH);

  leftWheelServo.attach(leftWheel);
  rightWheelServo.attach(rightWheel);
}

//Interupt Service Routine
//Run whenever interupt 0,1,2,3 fires as attached in setupMotorControl() to the
//  different wheel encoder //inputs
//Implemented the quad interface from the following
//  website: http://mkesc.co.uk/ise.pdf
void readencoder()
{
  //Pin2 is mapped to PE4, Pin3 is mapped to PE5
  currLeft = B00110000 & PINE;
  //Pin21 is mappeed to PD0, Pin20 is mapped to PD1
  currRight = B00000011 & PIND;
  //Shift currLeft variable to Bit0 and Bit1
  currLeft = currLeft >> 4;
  //Shift prevLeft value to Bit2 and Bit3
  prevLeft = prevLeft << 2;
  //Combine currLeft and prevLeft into one variable used to determine
  encIn = currLeft | prevLeft;
  //what value must be added to encCntLeft
  encCntLeft += encoderArray[encIn];
  prevLeft = currLeft;

  prevRight = prevRight << 2;
  encIn = currRight | prevRight;
  encCntRight += encoderArray[encIn];
  prevRight = currRight;
}
```

Listing A-6: Velocity control implementation

```
void velocityControl(float v1, float w1)
{
  //PID_dc --> It is the duty cycle for the updates send to the PID controller.
  //Ticks_l and ticks_r must be multiplied with this value to calculate the total
 ticks per second
  ticks_l = (encCntLeft - prevEncCntLeft) * PID_dc;    //Calculates the ticks per
 //second for left wheel
  ticks_r = (encCntRight - prevEncCntRight) * PID_dc;  //Calcualtes the ticks per
 //second for the right wheel

  float dotOmega_L = (v1 - wheelbase/2*w1)/wheel_circ;    //Embedded Robotics, Ch
//p 8.6, p 143, Inverse Kinematics
  ticks_desired_l = dotOmega_L * ticks_per_rev;
```

```
  float dotOmega_R = (v1 + wheelbase/2*w1)/wheel_circ;
  ticks_desired_r = dotOmega_R * ticks_per_rev;

  error_l = (ticks_desired_l - ticks_l);
  error_r = (ticks_desired_r - ticks_r);

  //Left motor PID Controller
  l_mot = Kp * (error_l - e_old) + Ki*(error_l + e_old)/2 + Kd*(error_l-
2*e_old+e_old2);
  //Right motor PID Controller
  r_mot = Kp * (error_r - e_old_r) + Ki*(error_r + e_old_r)/2 + Kd*(error_r-
2*e_old_r+e_old2_r);
  //Distance travelled between wheels PID controller
  error_dist = ticks_l - ticks_r + w1;
  error_mot = Kp_speed * (error_dist - error_dist_old) + Ki_speed * (error_dist +
 error_dist_old)/2;
  //Control signals for left and right motor
  l_control += l_mot - error_mot;
  r_control += r_mot + error_mot;

  //Clamps control signals to accepted values
  if (l_control > 2000) l_control = 2000;
  if (l_control < 1000) l_control = 1000;

  if (r_control > 2000) r_control = 2000;
  if (r_control < 1000) r_control = 1000;

  leftWheelServo.writeMicroseconds(l_control);
  rightWheelServo.writeMicroseconds(r_control);

  prevEncCntLeft = encCntLeft;
  e_old2 = e_old;
  e_old = error_l;

  prevEncCntRight = encCntRight;
  e_old2_r = e_old_r;
  e_old_r = error_r;

  error_dist_old = error_dist;
}
```

Listing A-7: Dead reckoning implementation

```
s_l = 2*pi*wheel_radius * ticks_l / PID_dc / ticks_per_rev;   //Must devide by PI
D_dc in order to get real delta value
s_r = 2*pi*wheel_radius * ticks_r / PID_dc / ticks_per_rev;
s = (s_l + s_r) / 2;
delta_phi = (s_r - s_l) / wheelbase;    //+w = counter clockwise

delta_x = s*cos(robotState[2]);
delta_y = s*sin(robotState[2]);
robotState[0] += delta_x;
robotState[1] += delta_y;
robotState[2] += delta_phi;
```

Listing A-8: Ultrasonic sensor polling implementation

```
1.  //Code for ultrasonic SRF02 modules on an Arduino Mega board
2.  //All sensors are in series using I2C and will be polled when distance data is
3.  //needed
4.
5.  //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
6.  //Sensors must be pre-programmed with unique IDs
7.
8.  #define CMD_REAL_RANGE_TX_cm      84
9.  #define CMD_REAL_RANGE_NO_TX_cm   81
10. #define CMD_GET_RANGE             94
11.
12. //## Defines the serial ports and baud rates used by the program
13. #define PCComms Serial
14. #define PCBaud 9600
15. #define USComms Serial2
16. #define USBaud 9600
17.
18. //### Time variables used to ping each ultrasound sensor
19. unsigned long int oldTimePing = 0;
20. int deltaPing = 49;
21.
22. //### Time variables used to timewhen new data must be sent serially
23. unsigned long int oldTimeTX = 0;
24. int deltaTX = 299;
25.
26. unsigned long int time = 0;
27. #define TimeOut 50
28. //##Delay time used to wait before next sensor is pinged to get distance data
29. #define pingDelay 20
30.
31. //## This array holds the adresses of the sensors in the sequence with which each
     one will be pinged
32. int sensorAddr[] = {3,0,4,1,5,2,6};
33. //## The function SIZEOF gives the amount of bytes used in the array and not the
    actual elements.
34. //##    When using int every element is 2 bytes long therefore the value must be
    divided by the siezeof(int)
35. //##    to get the total amount of elements in the array
36. const int numSensors = sizeof(sensorAddr)/ sizeof(int);
37. unsigned int sensorDist[numSensors];    //Array used to store the values of the d
    istances measured
38. int sensorCnt = 0; //Cntr used to keep track of the sensors ping'ed
39. unsigned int cntr = 0;
40.
41. void setup()
42. {
43.   //## This serial port will be used to communicate with the algorithm layer
44.   PCComms.begin(PCBaud);
45.   PCComms.println("PC comms started:");
46.   //## Starts the serial port connected to the SRF sensors
47.   USComms.begin(USBaud);
48.   //## Sends OK signal to PC after USComms started
49.   PCComms.println("US Serial port Started:");
50. }
51.
52. void loop()
53. {
54.   for (int n = 0; n < numSensors; n++)
55.   {
56.     delay(pingDelay);   //This delay ensure that ultrasonic waves disipated enoug
    h before pinging the next sensor, might be changed with millis()
57.     ping (sensorAddr[n]);
58.   }
```

```
59.
60.   String outData = "d";
61.   for (int n = 0; n < numSensors; n++)
62.   {
63.     int tmpRange = getRange(n);
64.     outData += n;
65.     outData += ":";
66.     outData += tmpRange;
67. //Do not add a comma after the last sensor data is added to the string
68.     if (n != numSensors-1)
69.     {
70.       outData += ",";
71.     }
72.   }
73.   PCComms.println(outData);
74. }
75.
76. ////////////////////////////////////////////////////////////////////////////
77. // Funtions and Procedures start here
78.
79. //### Tasks a sensor in calculating range and holding range data in buffer
80. void ping(int _sensorAddr)
81. {
82.    USComms.write(_sensorAddr);
83.    USComms.write(CMD_REAL_RANGE_NO_TX_cm);
84. }
85.
86. //###Reads range data from sensor buffer
87. int getRange(int _sensorAddr)
88. {
89.   static unsigned long timeLastInput = 0;
90.   unsigned long now;
91.   bool boolTimeOut = false;
92.   unsigned int rxData = 0;
93.
94.   USComms.write(_sensorAddr);
95.   USComms.write(CMD_GET_RANGE);
96.
97.   timeLastInput = millis();   //Set variable to current time before going into
98.                               //  the timeout loop
99.   while ((USComms.available() < 2) && (!boolTimeOut))
100.         {
101.           now = millis();
102.           if (now - timeLastInput > TimeOut) boolTimeOut = true;
103.         }
104.
105.         if (USComms.available() == 2)
106.         {
107.           rxData = USComms.read() << 8;
108.           rxData |= USComms.read();
109.         }
110.
111.         return rxData;
112.       }
```

Appendix B

Indonesian Journal of Electrical Engineering and Computer Science
Vol. 15, No. 2, August 2019, pp. 1095~1101
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v15.i2.pp1095-1101

Automated library booktruck for traditional libraries

J. J. Spies[1], B. Kotze[2]
[1]Department of Computer Systems Engineering, Tshwane University of Technology, South Africa
[2]Department of Electrical, Electronic and Computer Engineering, Central University of Technology, South Africa

Article Info

Article history:

Received Sep 8, 2018
Revised Feb 10, 2019
Accepted Feb 25, 2019

Keywords:

Autonomous vehicle
Library
Localization
Path-finding
Simulation

ABSTRACT

Libraries are an integral part of our society's knowledge repository and even though technological advances such as the internet, smart devices and an 'always-connected-society', provide avenues for fast and almost instantaneous access to knowledge, libraries still provide a physical place for the collection and dissemination of knowledge. The prompt shelving of the physical returned library books is an important task in any traditional library. To help speed up the shelving process, this paper proposed and simulated an automated booktruck that is capable of moving returned library books from the return desk back to the shelves. The simulation models currently available robotic hardware and implemented path finding and localization. The simulation results showed that returned books can be delivered to the shelves four times faster than the by using the current practices.

Corresponding Author:

J. J. Spies,
Wireless and Radio Science Centre,
Department of Computer Systems Engineering,
Tshwane University of Technology,
South Africa

1. INTRODUCTION

The process of physically moving returned library books from the lending desk back to their respective shelves and physically shelving the books, is an important albeit tedious process in any library. In a traditional library, when a book is returned, it is scanned into the Library Information Systems (LIS) and placed on a booktruck or other staging area, to be shelved at the convenience of the library staff, or using the protocol implemented by the library.

Studies done by the University of Virginia and Bryan College have shown that library books take between 1 and 5 days to travel from the return desk to their relevant shelves. The study from the University of Virginia also showed that book pick-ups are done more frequently from the most obvious places, such as shelve ends, and less frequently from the less obvious places, such as study cubicles [1]. This delay shelving the books and picking up the books, leads to books showing as available in the library database but which are not available on the shelves, in turn leading to frustration amongst users since books cannot be found [2].

A study done by [3] on shelving concluded that shelving is a repetitive and physically demanding task, especially when trolleys full of books need to be moved from the circulation desk to the relevant shelves. This physically demanding task can be alleviated by implementing service robots [4].

The results of this study show that it is possible for a service robot to move books from the lending desk back to a collection spot in a short enough time to significantly reduce material dead time. By implementing an automated booktruck having this automated booktruck, library material dead time will be shorter and the physically demanding task of pushing a heavy booktruck is eliminated.

2. LITERATURE REVIEW

Robotics in libraries are not something new. The first robot used in a library was in a Swedish library [5] at the lending desk. Books were placed on a conveyer system from where the robotic arm would identify each book and place it in a dedicated slot for later shelving. Since the introduction of robots and technology in libraries, research continues in the following areas.

2.1. Inventory Control

Inventory control and finding books when they are already shelved, using shelf scanning methods which incorporates radio frequency identification (RFID) tags, placed inside the books, and/or barcode numbers attached to the back of books [6-9].

2.2. Book manupilation

Placing books back into shelves or retrieving books from shelves is termed as book manupilation. Purpose built grippers attached to robotic arms [10, 11], and also teleoperation interfaces for retrieving books from shelves or storage locations, with the capability of opening and paging through the book while viewing the page content [12] is part of the current research.

2.3. Library Asistants

Hugh is the latest iteration of library assistant robots [13]. Hugh will be able to take verbal commands and then take the library user to where the specified book can be found. A framework for simplifying book finding in a children's library was developed by [14], while [15] developed a library assistant capable of assisting the elderly and other individuals to find library resources quicker.

3. MOTIVATION FOR AUTOMATED BOOKTRUCK

In contrast to the previous research, the authors of this paper have identified that no-one has looked at an automated way of moving books from the lending desk back to the shelves and therefore this paper proposes an automated booktruck which will identify books using embedded RFID tags and move books from the lending desk back to central collection points amongst the shelves. These collection points are then visited by shelvers from where the returned books are manually shelved into the correct position. No attempts will be made to shelve the books.

4. SIMULATION OVERVIEW

To test the hypothesis that an automated booktruck will reduce material dead time in traditional libraries, a simulation of an automated platform was performed using the library map of the Tshwane Univeristy of Technology's, Soshanguve South campus.

The simulator was written in Processing, a flexible software sketchbook and language for learning how to code within the context of the visual arts. Since Processing is intended to promote visual literacy, creating and working with screen objects and graphics, implementation of these objects is uncomplicated, simple and easily achieved.

The hardware platform used as basis for the simulated model, is the Robot Base Kit (#28976, #28977) and the motor mount and wheel kit (#27971) with Position Controller (#29319) all of which were available from Parallax Inc. It was selected because it closely resembles the footprint of an average human being. It is also big enough to carry books to the drop-off points.

Seven sonar sensors were used to achieve localization with their secondary purpose being obstacle avoidance. These sensors cover the front of the robot. Figure 1 shows the placement of the ultrasonic sensors. To reduce crosstalk, distance detection is done implementing the following sequence: 3, 0, 4, 1, 5, 2, 6. A Microsoft Kinect Sensors were used as primary obstacle detector. The data received is converted into 2-dimensions which is then used to update the map of the environment.

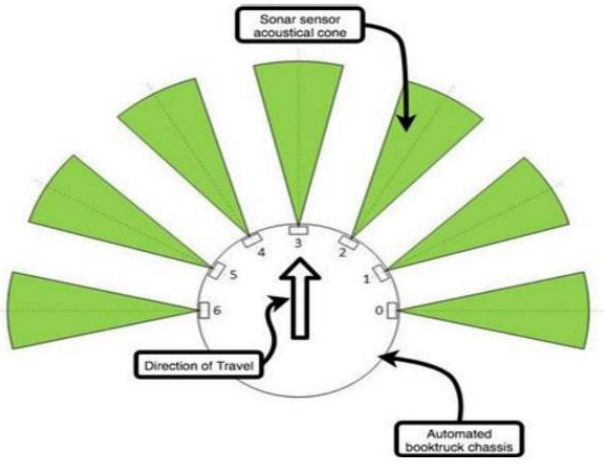


Figure 1. Ultrasonic Sensor Placement

5. SIMULATOR

Figure 2 shows a fictitious floorplan which highlights the conversion of the imported map into an occupancy grid. It also shows the node creation using the Quad-tree method, the robot position (green circle), the goal position (target sprite), the global route to follow (green line) the sonar sensor data used for localization (small white circles) and the best guess of the robot position using particle filters (red circles).

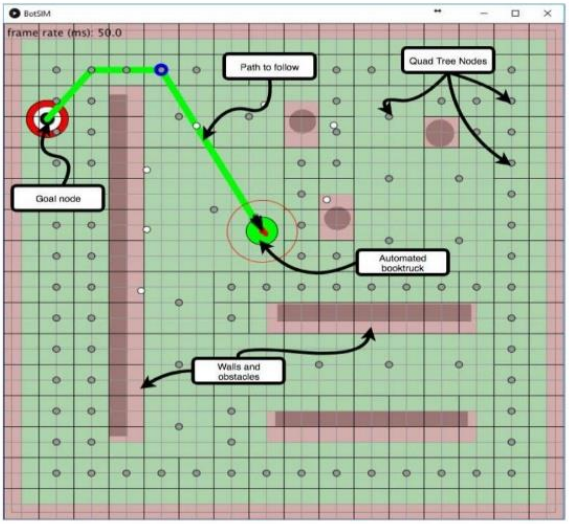


Figure 2. Example Floor Layout

5.1. Environmental map

The simulation was done using the floorplan and open collection area of the library at the Soshanguve South (SSoc) campus of the Tshwane University of Technology (TUT). Figure 3 shows the ground floor of the library while Figure 4 shows the floorplan, converted into an occupancy grid map during simulation.

This floorplan can be created by scanning a floor layout of the environment or by drawing the floor layout using CAD software or similarly suited software able to export a drawing as a JPG or PNG file. The dimensions of the exported picture are used to correctly determine the scale of the objects on the map and to place detected obstacles on the map.

When the simulator starts it will import the map and divide the map into an occupancy grid equivalent map. The tiles in this occupancy grid keeps data like walls, obstacles, etc. which is used by the path planning phase to determine the best path to the selected drop-off point.

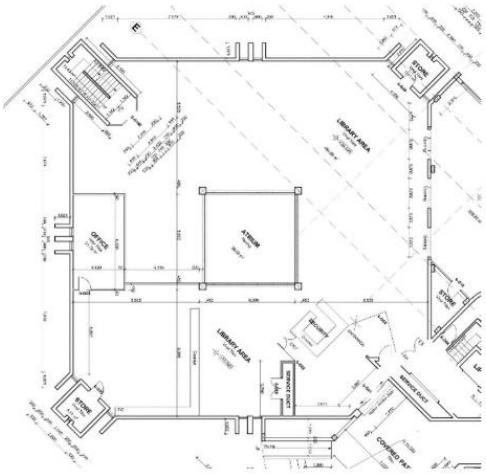


Figure 3. Library Floor Plan

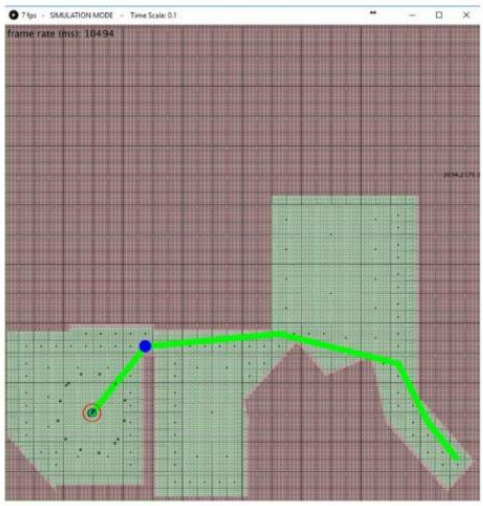


Figure 4. SSoc Ground Floor

5.2. Path Planning

Global path planning is achieved using the A-star path planning algorithm [16]. The path starts at a START node and ends at a GOAL node. The START node is the current booktruck position and the GOAL node the drop-off point for the book. The path is constantly being updated using data form the map.

5.3. Local Path Planning

Local navigation is achieved by using the potential field method where a pushing and pulling forces are combined to movement vectors. The pulling force is created by the next node in the path towards the goal position. Pushing forces are created away from all the obstacles. Two types of obstacles exist: 1) Fixed obstacles which is part of the initial map, for example: walls, book shelves and other fixtures on the floor of the environment, and 2) Random obstacles which are obstacles detected by the Kinect Sensor. A decay value is attached to random obstacles. This decay value removes the obstacles from the map after it expired to simulate the possible movement of random obstacles.

By combining the pushing and pulling forces, a vector, representing the speed and direction the robot should travel in, is created. Applying this vector to the robot has the robot reach the goal while avoiding any immediate obstacles. Any obstacles detected, while the booktruck moves, are added to the map. These new additions to the map influence the potential field, leading to new routes and movement patterns.

5.4. Localization

The purpose of localization is to estimate the robot pose using sensor data. When localized, the robot will be able to determine the optimized path between its current location and the next node position in the path towards the goal location. Localization is achieved by using simulated odometry and sonar data. Both data sets had noise values added to simulate real world conditions.

6. RESULTS

For this paper, an existing mobile platform was simulated using the floorplan of the SSoc at TUT to determine if it is possible for an automated booktruck to return books to the shelves in less time. The lending desk is on the ground floor while the open shelves are on the first floor. To overcome this problem, the simulation was divided into two parts. The first part simulated the robot moving towards the stairs on the ground floor, and the second part of the simulation measured time taken from the stairs' landing on the first floor to the relevant drop off point between the shelves. These two times were added together to produce a

total time in one direction. This time was then multiplied by two to provide for the robot to return to the lending desk.

The data retrieved from the Library Information Systems (LIS) is graphically depicted in Figure 5. This graph shows the number of books returned per month for the year, 2016.

Figure 5. Book returns per month for the year 2016

This data set was then converted into a time value shown in Figure 6. This minute value represents the maximum time available to return a book to the shelve. This value was calculated using the monthly data from Figure 5 and assuming a total of 21.8 working days per month, with a typical workday consisting of 8 hours.

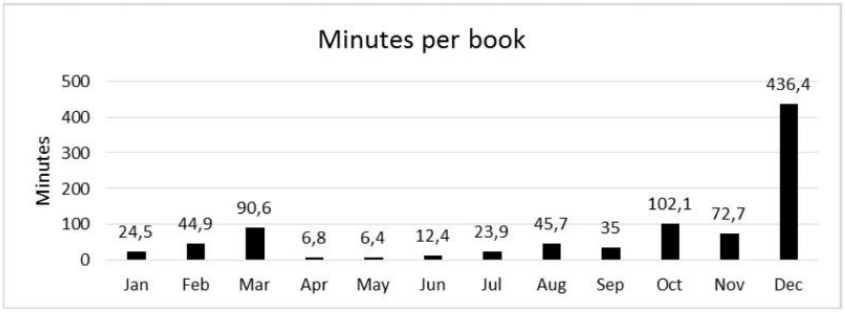


Figure 6. Time taken to shelve a single book

The maximum speed of the simulated booktruck was set at only 100cm/s. The starting position of the robot randomly picked within a 1m by 2m area behind the lending desk, this simulates an inexact starting point after returning from the shelves. The simulation was done a total of ten times for each of the floors. These values, which consisted of distance and time, was then added to determine a total time and a total distance covered as shown in Table 1.

Table 1. Results of Simulated Data

Lending Desk to Stairs		Stairs to drop off bin		Total Return Data	
Distance (cm)	Time (s)	Distance (cm)	Time (s)	Distance (cm)	Time (s)
2546.433	25.7	1867.719	19	8828.304	89.4
2544.989	25.6	1864.812	18.8	8819.602	88.8
2479.516	24.9	1868.632	19.1	8696.296	88
2464.285	24.8	1864.812	18.8	8658.194	87.2
2487.262	25	1866.515	18.9	8707.554	87.8
2424.416	24.4	1866.515	18.9	8581.862	86.6
2493.986	25.1	1864.812	18.8	8717.596	87.8
2415.776	24.4	1864.812	18.8	8561.176	86.4
2487.811	25.1	1867.719	19	8711.06	88.2
2534.231	25.5	1866.515	18.9	8801.492	88.8
		Averages		8708.3136	87.9

An average time of the total times was calculated and then compared to the values from Figure 6 to determine if the booktruck would be faster. In all the values calculated for each month, the booktruck time was less by at least a factor of 4. The following conclusions were made when analyzing the data:

- Based on the total return time, an automated booktruck will be able to return books to the designated drop off points well within the calculated book return times as shown in the Figure 6.
- The SSoc is a small library and data might look different in large and busy libraries.

7. FUTURE WORK

Although Processing was used to create the simulator, it is meant to be an IDE for quick sketches and to test smaller concepts and ideas. Even though the Processing IDE is capable of working with multiple files, the lack of robust debugging tools is a problem when implementing large projects. In future, the design will be moved over to Robotic Operating Systems (ROS) which will be used to verify the found results.

The Microsoft Kinect sensor is a proven sensor when it comes to robotics and automated vehicle navigation, however newer sensors have become available and ideally it must be replaced by the Intel RealSense range of sensors or a LIDAR range sensor.

Research can be conducted regarding a method of handling multiple pieces of library material at the same time. This might be achieved by adding a mechanism capable of identifying and handling more than one item or by having a separate dispensing unit for when the autonomous booktruck returns from the shelves, for example an adapted conveyer belt system based on [5]. This system might then automatically load the next book onto the booktruck. Together with multiple books, an AI implementation of route optimization can be employed to optimize the delivery of multiple books.

Informal discussion with library staff, when conducting measurements to verify the floorplan, revealed a concern regarding automation robbing staff members of jobs. Automation used to make one's life easier was not seen as an issue if it assisted people and did not rob them of jobs.

REFERENCES

[1] M. Mitchell, "Library Workflow Redesign: Concepts and Results," *Library Workflow Redesign: Six Case Studies*, vol. 139, pp. 1, 2007.
[2] C. Cook and F. M. Heath, "Users' perceptions of library service quality: A LibQUAL+ qualitative study," *Library Trends*, vol. 49, pp. 548, 2001.
[3] F. N. Onifade, *et al.*, "Staff Attitude to Shelving and Shelf Reading in Academic Libraries," *North Carolina Libraries*, vol. 68, pp. 12, 2010.
[4] H. Moradi, *et al.*, "Service robotics (the rise and bloom of service robots)," *Robotics & Automation Magazine*, vol. 20, pp. 22-24, 2013.
[5] R. Hansson, "Robot lends a hand in a Swedish library," *Industrial Robot: An International Journal*, vol. 22, pp. 34-35, 1995.
[6] R. Li, *et al.*, "AuRoSS: An Autonomous Robotic Shelf Scanning System," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 6100-6105, 2015.
[7] M. K. Sinha and A. Chanda, "Exploring RFID Technology Application for Managing Library and Information Services in University and Institutional Libraries of North East India: An Overview," *Sinha, Manoj Kumar and Chanda, Anupam (2014). Exploring RFID Technology Application for Managing Library and Information Services in University and Institutional Libraries of North East India: An overview. International Journal of Information Sources and Services*, vol. 1, 2014.
[8] M. Rashid, *et al.*, "Sorting and Retrieval Robotic System Controlled via Programmable Logic Controller for Library Usage," *International Journal of u-and e-Service, Science and Technology*, vol. 7, pp. 19-30, 2014.
[9] J. Thirumurugan, *et al.*, "Line following robot for library inventory management system," in *Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on*, pp. 1-3, 2010.
[10] M. Prats, *et al.*, "The UJI librarian robot," *Intelligent Service Robotics*, vol. 1, pp. 321-335, 2008.
[11] B. K. Kim, *et al.*, "Design and control of the librarian robot system in the ubiquitous robot technology space," in *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and IHuman Interactive Communication*, pp. 616-621, 2008.
[12] T. Tomizawa, *et al.*, "Book browsing system using an autonomous mobile robot teleoperated via the internet," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2, pp. 1284-1289, 2002.
[13] "Robot librarian designed by Aberystwyth University students," 2016. Available: https://www.timeshighereducation.com/news/robot-librarian-designed-aberystwyth-university-students.
[14] W. Lin, *et al.*, "Developing a service robot for a children's library: A design-based research approach," *Journal of the Association for Information Science and Technology*, vol. 65, pp. 290-301, 2014.
[15] J. Behan and D. T. O'Keeffe, "The development of an autonomous service robot. Implementation: 'Lucas'-The library assistant robot," *Intelligent Service Robotics*, vol. 1, pp. 73-89, 2008.