Development of an Autonomous Radiation Mapping Robot

by

Michael Edward Hosmar

Bachelor of Mechanical Engineering (Hons), University of Ontario Institute of Technology, 2014

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

Master of Applied Science

in

The Faculty of Engineering and Applied Science

Mechanical Engineering

Supervisor(s): Scott Nokleby, University of Ontario Institute of Technology
Ed Waller, University of Ontario Institute of Technology
Examinig Board: Remon Pop-Iliev, University of Ontario Institute of Technology
External Examiner: Andrew Hogue, University of Ontario Institute of Technology

THE UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

April 2017

©Michael Edward Hosmar 2017
Abstract

The development of an autonomous robotic platform for generating radiation maps is presented. An integrated autonomous exploration algorithm, Particle Swarm Optimization algorithm, and the ability to accurately localize and map multiple radiation sources in both indoor and outdoor environments using actual sources are features of the system presented.

Radiation maps provide an easy to understand view of the invisible hazards that radioactive sources pose. Previous methods of producing radiation maps either required prior knowledge of the physical dimensions of the mapping area or needed the intervention of a human operator. The method presented here improves on previous methods in that it does not require prior knowledge of the area or the intervention of an operator. The implementation consists of three main components: an exploration algorithm, a navigation system, and a source localization system. The exploration system guides the robot through the area instructing it to take radiation measurements as necessary. The navigation system provides accurate localization to maintain the accuracy of the measurements. The source localization system then uses the measurements and a radiation model to produce an estimate of the source positions and intensities. A live intensity heatmap displays preliminary information of the surroundings while the robot is exploring the area, providing useful information from the start of its operation. The intensity heatmap is updated while the robot explores, providing a more detailed view as the robot progresses through an area.

The technical details of the implementation and the experimental results of the radiation mapping capabilities of a fully Autonomous Radiation Mapping Robot (ARMR) are presented in detail here. Multiple scenarios are tested, both in an outdoor and indoor environment. In the scenarios an unknown configuration of Cesium 137 sources is explored and mapped by the ARMR. The results demonstrate the effectiveness of
the ARMR as a tool for mapping radiation sources in unknown environments. Such a system could be used for monitoring nuclear facilities or deployed in terrorist or radiation accident scenarios.
Acknowledgements

I would like to thank my supervisors Dr. Scott Nokleby and Dr. Ed Waller, their support and guidance made this work possible. I would like to thank my colleges in the Mechatronic and Robotic Systems Laboratory. You have all made my time in the lab an enjoyable one.

I would like to give a special thanks to Tanya Neretljak as well as Sharman Perera, Callan Brown, and everyone else who helped me arrange for the safe use and transport of the radiation sources.

Finally, I would like to thank the University Network of Excellence in Nuclear Engineering (UNENE) and the Natural Sciences and Engineering Research Council (NSERC) of Canada for their financial support of this research.
# Table of Contents

Abstract ii  
Acknowledgements iv  
Table of Contents ix  
List of Tables x  
List of Figures xiii  
Acronyms xiv  
Glossary xvi  

1 Introduction 1  
1.1 Radioactivity and Radiation Mapping ..................... 3  
1.2 Mobile Robotics ........................................... 6  
1.2.1 Locomotion ............................................. 7  
1.2.2 Sensors ..................................................... 8  
1.2.3 Navigation and Mapping ................................. 11  
1.3 Problem Statement and Requirements ..................... 13  
1.4 Summary of Contents ....................................... 15  

2 Background and Literature Review 17
2.1 Literature Review ................................................. 17
  2.1.1 Autonomous Exploration ................................... 17
    2.1.1.1 Boustrophedon Cellular Decomposition ............... 18
    2.1.1.2 Potential Field Distribution ......................... 18
    2.1.1.3 Entropy Model ........................................ 19
    2.1.1.4 Radiation Mapping With Entropy Modeling .......... 19
    2.1.1.5 Frontier Exploration ................................. 20
  2.1.2 Statistical Source Localization ............................ 21
  2.1.3 Radiation Mapping Robots .................................. 23
  2.2 Radiation Detection ........................................... 25
    2.2.1 Ionization Detectors .................................... 25
    2.2.2 Solid-state Detectors ................................... 27
  2.3 Robot Operating System (ROS) ................................ 29
    2.3.1 Nodes ...................................................... 30
    2.3.2 Launch Files and Parameters ............................. 31
    2.3.3 Communication Methods ................................ 32
      2.3.3.1 Topics ............................................... 32
      2.3.3.2 Services ............................................. 33
      2.3.3.3 Actions .............................................. 34
    2.3.4 Transform System ......................................... 34
    2.3.5 Robot Visualiser ......................................... 35
    2.3.6 Debugging Tools ......................................... 35
  2.4 Robot Localization and Mapping ............................... 36
    2.4.1 Pose Estimation .......................................... 36
    2.4.2 Filtering Techniques .................................... 38
      2.4.2.1 Complimentary Filter ............................... 38
      2.4.2.2 Explicit Complimentary Filter ...................... 39
3 System Overview

3.1 Functional Overview ........................................... 48
3.2 Hardware Overview ............................................. 50
  3.2.1 Robotic Base ............................................. 50
  3.2.2 External Sensors ........................................... 51
    3.2.2.1 LiDAR ............................................... 51
    3.2.2.2 Radiation Detector ................................... 52
3.3 Input Processing Nodes ......................................... 52
  3.3.1 Jackal Node ............................................... 53
  3.3.2 Robot State Publisher ..................................... 54
  3.3.3 LMS1XX .................................................. 54
  3.3.4 GPS Processing ........................................... 54
  3.3.5 Robot Localization Node ................................... 55
  3.3.6 Command Velocity Multiplexer ............................. 57
  3.3.7 Bluetooth Node ........................................... 58
4 Navigation, Exploration, and Source Localization 60

4.1 Navigation and Localization Subsystem 61

4.1.1 Navigation 61

4.1.1.1 Indoor Navigation 62

4.1.1.2 Outdoor Navigation 63

4.1.2 Localization 64

4.1.2.1 Robot Localization Node 64

4.1.2.2 Multiple Robot Localization Nodes 65

4.1.2.3 GPS Tests 66

4.1.2.4 RTK GPS 69

4.1.2.5 GPS Processing Tests 69

4.2 Exploration Subsystem 71

4.2.1 Indoor Exploration 71

4.2.2 Outdoor Exploration 72

4.2.3 Control Node Autonomous Sampling 74

4.3 Radiation Detection Subsystem 77

4.4 Source Localization Subsystem 80

4.4.1 Problem Statement 80

4.4.2 Cost Function 81

4.4.3 PSO Method 82

4.4.4 Testing and Modifications to Base Algorithm 83

4.4.5 ROS Integration 86

4.5 Displays and Controls 87

4.5.1 Graphical User Interface 87

4.5.2 Live Radiation Intensity Map 90

5 Experimental Results and Discussion 93

5.1 Indoor Tests 93
List of Tables

5.1 A comparison of the time taken to explore the area using different numbers of samples. ........................................... 98
5.2 A comparison of the time taken against different number of sources and samples for the source localization algorithm. ......................... 99
5.3 A table displaying the Root Mean Square (RMS) error of each test when compared to the reference location. .............................. 100
5.4 Source intensity predictions for the indoor trials. ............................ 101
5.5 The UTM positions of the surveyed locations of the source locations and base location. ......................................................... 110
5.6 The time taken to search a 20 m square area. ............................... 113
5.7 The RMS error of each test when compared to the Global Positioning System (GPS) reference. .................................................. 117
5.8 The source intensities reported by the source localization algorithm for the outdoor tests. ......................................................... 118
List of Figures

1.1 A robot used at the Fukushima Daiichi nuclear site. . . . . . . . . . . . 3
1.2 An outdoor mobile robotic platform. . . . . . . . . . . . . . . . . . . . 7
1.3 An example of a Micro Electro-Mechanical System (MEMS) accelerometer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
1.4 An example of a Light Detection And Ranging (LiDAR) sensor. . . . 12

2.1 Frontier-based exploration of an office. . . . . . . . . . . . . . . . . . 21
2.2 Principle of operation of a GM tube. . . . . . . . . . . . . . . . . . . 27
2.3 Comparison of different scintillation compositions. . . . . . . . . . . 28
2.4 2D Comparison of Extended Kalman Filter (EKF) and UKF. . . . . . 41

3.1 A functional view of the Autonomous Radiation Mapping Robot (ARMR). 49
3.2 The Jackal robotic platform by Clearpath Robotics. . . . . . . . . . . . 50
3.3 The bracket made to attach the NaI radiation detector. . . . . . . . . 53

4.1 An overview of the interactions of the “move_base” package. . . . . 61
4.2 The connections between the two different localization nodes (ekf_localization and global_localization) used on the ARMR and their interfaces. . . 64
4.3 Position estimates of a stationary GPS Sensor stationed in the Polonsky Commons, University of Ontario Institute of Technology (UOIT). . . 67
4.4 Position estimates of a stationary GPS Sensor stationed in a parking lot at UOIT. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
4.5 Position estimates of a stationary Real Time Kinematic (RTK) Sensor stationed in a parking lot. Here the $y$ axis is the $x$ position of the detector and the $x$ axis is time in seconds. ................. 70

4.6 The exploration plan before (a) and after completion (b). ........ 73

4.7 Control node basic algorithm. .......................... 75

4.8 The URSA Multi Channel Analyser (MCA) software. ............ 78

4.9 A comparison of the Markov chain Monte Carlo (MCMC) results from McDougall et al. and the results of different neighborhood topologies used in the Particle Swarm Optimization (PSO) algorithm. .......... 84

4.10 The Graphical User Interface (GUI) used on the ARMR. ........ 87

4.11 An example of using the ArealMapDisplay display type with the ARMR. 90

5.1 A view of the test area for indoor testing. ....................... 94

5.2 The setup used for configuration 1. ............................ 95

5.3 The setup used for Configuration 2. ............................ 96

5.4 Indoor exploration operation for different sample spacings. ....... 102

5.5 Estimated source locations for Configuration 1 with 1.3 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red stars). ......................... 103

5.6 Estimated source locations for Configuration 1 with 1.0 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red stars) .............................. 104

5.7 Estimated source locations for Configuration 1 with 0.75 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red stars) ......................... 105

5.8 Estimated source location for Configuration 2 with 1.3 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red star). ............................ 106
5.9 Estimated source location for Configuration 2 with 1.0 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red star) .................................................. 107
5.10 Estimated source location for Configuration 2 with 0.75 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red star). .................................................. 108
5.11 A view of the command centre and base station location near home plate. 110
5.12 A view of the two source positions inside the baseball diamond. ..... 111
5.13 Sample locations determined by exploration algorithm for Configuration 1 (a,b) and Configuration 2 (c,d). .................................................. 114
5.14 Estimated source locations produced by the PSO algorithm (crosses or squares) with the GPS reference inserted (red stars). ...................... 115
5.15 Sample locations determined by exploration algorithm. ................. 116
5.16 The GPS reference location compared to the live view of Configuration 2, Test 2. ................................................................. 118
5.17 An example of the GUI view during a test ................................... 120
Acronyms

AMCL  Adaptive Monte Carlo Localization.

API  Application Program Interface.

ARMR  Autonomous Radiation Mapping Robot.

CPS  Counts Per Second.

DAC  Digital Analogue Converter.

EKF  Extended Kalman Filter.

EMI  Electro-Magnetic Interference.

GPS  Global Positioning System.

GUI  Graphical User Interface.

IMU  Inertial Measurement Unit.

LiDAR  Light Detection And Ranging.

MARG  Magnetic Angular Rate Gravity.

MCA  Multi Channel Analyser.
MCL  Monte Carlo Localization.

MCMC  Markov chain Monte Carlo.

MEMS  Micro Electro-Mechanical System.

PPP  see Precise Point Positioning.

PSO  Particle Swarm Optimization.

RMS  Root Mean Square.

ROS  Robot Operating System.

RTK  Real Time Kinematic.

SLAM  Simultaneous Localization And Mapping.

UGV  Unmanned Ground Vehicle.

UOIT  University of Ontario Institute of Technology.

URDF  see Universal Robot Description Format.
Glossary

**Costmap**  A costmap or occupancy grid is a method for describing the traversability of an area using a pixel map, where each pixel is given a value which ranges from empty and therefore traversable to occupied or untraversable.

**Message**  In the ROS framework it is a predefined variable of basic data type or a set of variables or other messages or any combination of such. This message type must be defined at compile time and is used to communicate other nodes through publishers and subscribers.

**Node**  Is a program which runs in the ROS framework and can use ROS messages to interact with other nodes.

**Pose**  The combination of a position and orientation vector.

**Precise Point Positioning**  A method for accurately determining a position using a single global positioning sensor.

**Publisher**  Is used in a ROS node to send a filled message to any subscribers.

**Rviz**  A node in ROS for displaying and interacting with live data from a robot.

**Subscriber**  Is used in a ROS node to receive a filled message from a topic which has been published to.
**Topic** Is a named communication stream which can be published or subscribed to.

   A topic may only have one message type.

**Universal Robot Description Format** A method for pragmatically describing the shape and linkage configuration of the robot as well as associating graphical meshes to the model.
To my parents, this work would not have been possible without your continued support and encouragement.
Chapter 1

Introduction

Nuclear energy has cemented itself as a large producer of electric energy across the world. It provides clean and reliable power while reducing the need for green house gas producing electric energy production such as coal or natural gas. In places such as the Far East the nuclear energy production is booming having 30 reactors under construction [1]. Nuclear energy however has its inherent risks and with them the possibility to cause lasting harm to any areas affected. Safety should be a high priority in the path of development for the nuclear industry. This is especially true for emergency responders who will be faced with the most risky situations. Dealing with emergency situations in a safety conscious way can be difficult because they are often unpredictable and unforeseen factors often arise. Unknown hazards present the most risk in any situation. Due to radiation’s invisible nature it becomes difficult to both recognize and locate sources of radiation. An autonomous radiation mapping robot could be very useful to both locate radioactive sources and assess the level of danger.

Emergency situations are hard to plan for due to their chaotic nature. Response protocol and emergency training does not remove people from potentially being exposed
to dangerous situations in order to do necessary work. These personnel rely on personal radiation detectors in order to navigate through the radioactive area unaware of the possible danger that could be in the next room. Working and moving through this environment is very difficult when one has to consider the risk of every step. Reacting to sudden changes in the environment is even more difficult if the dangers around the person are not fully known. Even in lower radiation environments it would be beneficial to know what the dose rates are throughout the work area and additionally what paths should be taken to minimize total dose.

An autonomous robot would be very beneficial in situations where personnel may be exposed to an unknown radiation environment. Robots have already been used in the nuclear industry for some time. However, they are often only used in emergency situations as a backup to personnel when radiation doses are too high or when the area is difficult to enter [2]. These existing systems are almost always tele-operated as well, relying on the skills of the operator to perform the task and to be as thorough as possible. An autonomous radiation mapping robot could prove useful as a first response option. This type of robot would need to measure radiation, autonomously navigate, produce informative information about the radioactive environment, and also, be easy to use. This radiation robot, being essentially a mobile sensor and aware of its location, would be able to use its data to not only produce an overall map of every radiation measurement but be able to process this information further to create a mathematically refined view of the environment.

In this thesis, the focus will be on the development of an ARMR as described above. An existing robotic base will be used as well as an existing radiation detector. The goal of the research is to develop the necessary software in order for the robot to complete all the tasks described above, as well as the creation of the software required to control, coordinate and monitor each of the robot subsystems. In order to completely understand the problem addressed by this thesis some additional background
information must be understood. Information about radiation mapping and mobile robotic sensing and mapping will be covered in the next sections.

1.1 Radioactivity and Radiation Mapping

Radioactive materials are naturally unstable atomic compounds which release energy in the form of energetic particles. There are three common types of radioactive decay characterized by the type of particle emitted. Alpha or $\alpha$-decay is the result of releasing two protons and two neutrons, effectively a He-4 atom. Beta or $\beta$-decay is the release of an electron or positron. Finally, gamma or $\gamma$-decay is the release of a highly energetic photon from the atom. The frequency of emission (called activity), type, and energy of radioactive emissions depends on the isotope and its properties. A detailed look at radiation and its mechanisms can be found in [4].

Alpha and beta particles are heavier, more charged and slower than gamma rays. This makes them much more susceptible to interacting with the environment and, therefore, of being stopped. This makes them much easier to shield from and limits
their risk of exposure to the process inhalation or ingestion. They are also less useful for mapping purposes because there is a high chance that they will be blocked by minor obstructions. Gamma rays in contrast have higher energy and are much less likely to interact with materials allowing them to pass through many objects. Their high energy means that they have the ability to ionise the material that they finally interact with which causes serious damage to living tissue. Gamma rays are usually transmitted along with alpha or beta rays, this fact, along with gamma ray’s high transmittance rate make them a good candidate for sensing in a radiation mapping scenario.

In order to map out radioactive sources, some predictability of the radioactive decay is needed. While it is true that radioactive decay is quite random and the interaction of a gamma ray before reaching the detector is complex, some statistical reasoning can be applied to a static environment. In general, the chance that a gamma ray will be emitted, travel in the direction of the detector, and interact with it, is modeled by the Poisson Distribution. However, this can be simplified further to a Gaussian Distribution when either the activity is high or the sample is taken over a relatively long period. Radiation sources which are of interest to radiation mapping will have high atomic numbers and therefore high specific activity as well as relatively high masses causing the sources to have overall activities much higher than background. This leads to the fact that the average number of gamma rays interacting with the detector remains constant in a static environment. The chance of a of radioactive decay in the next time period is constant in this case. This simplifies the process of determining the average number of gamma rays travelling through the detector which will help with localizing or determining the location of a source.

Radiation detectors produce two types of signals: Counts, (in Counts Per Second (CPS) or counts per minute) or, a spectrum of the different energies of each interaction. The simpler CPS is calculated based only on the source’s activity and some
parameters of the system. First, the geometry of the source, any shape other than a perfect sphere will produce concentrations of gamma rays in the direction of its convex surfaces. The other parameters depend on the detector. The detectors makeup and sensitivity to gamma rays affect the probability that a passing ray will be sensed and counted. The number of gamma rays that will actually pass through the detector is based on the environmental geometry and the detector’s geometry. Assuming that the source is relatively small compared to the distance to the detector it can be assumed that gamma rays will be emitted in all directions equally. Given a source that produces $S_o$ gamma rays per second at a distance $r$ centimeters from the detector, the equation becomes:

$$R \propto \frac{S_o}{4\pi r^2}$$

(1.1)

where $R$ is the radiative flux in $(\text{gammas/cm}^2\text{s})$. In order to find the number of CPS that the detector would actually report, the flux $R$ would be multiplied by the detector efficiency. It is clear to see in Equation (1.1) that keeping all parameters the same and only changing the radius results in a flux change of $\frac{1}{r^2}$. This simplification can be leveraged later when a sensor model must be developed for localizing sources.

Equation (1.1) describes how to predict the flux given the direction and strength of the source. However, this is leaving out one very important aspect. Any material in the path of the gamma rays has a chance of interacting with them and will affect the number of particles reaching the detector. Even air will provide some shielding. Air, however, is not dense enough to have a noticeable effect for a radiation mapping scenario. The amount of shielding a material will provide depends on two parameters: The thickness of the material in $cm$, and the materials attenuation factor in $cm^{-1}$. The attenuation factor is based on the material’s composition. Materials with a higher atomic number and which are denser will provide more attenuation. The attenuation
equation is as follows:

\[ e^{-\mu x} \]  

(1.2)

where \( \mu \) is the attenuation factor and \( x \) is the path length through the material. It is apparent in Equation (1.2), that as the length \( x \) increases the total value decreases approaching zero. Applying this to Equation (1.1), the total flux will decrease with increasing \( x \) and the equation becomes:

\[ R \propto \frac{S_0 e^{-\mu x}}{4\pi r^2} \]  

(1.3)

If there were multiple shielding materials with different thicknesses each material could have its own attenuation factor as Equation (1.2) and then each added up and multiplied by Equation (1.1). Using Equation (1.3) it is possible to determine the flux due to one source with shielding in the path. To determine the flux due to two or more sources the solution is only slightly different. Each source will have its own strength, radius, and possible shielding in its path to the detector. The result is that each source produces its own flux at the detector and the total flux will be the sum of each source. Assuming a detector with equal efficiencies in all directions this would also result in the CPS being the sum of each source. This knowledge will be useful when attempting to localize multiple sources.

1.2 Mobile Robotics

Mobile robotics concerns any robotic system that has locomotion. Specifically, for an ARMR, wheeled, ground robotics will be utilized. A mobile robot, like any robot, must contain sensors and actuators in order to fulfill a task automatically. A mobile robot’s primary task is to always produce an accurate representation of its current pose and its surroundings and to be able to traverse its environment.
1.2.1 Locomotion

There are many types of ground robotic locomotion from wheels to tracks to more exotic legged robots. Each has its benefits and weaknesses. Wheeled locomotion is very efficient and simple to control, but struggles with obstacle traversal. Even with wheeled robotics there are many different options. There is differential steering, where wheels on either side of the platform turn at different speeds and display tank like movement. The robot in Figure 1.2 is an example of a platform with differential steering. This type of configuration benefits from simplicity of control and design but suffers from inefficiencies when turning and instability at high speeds. Second, there is Ackermann or car-like steering. This configuration displays good high-speed stability and efficiency but is complex and difficult to control. Finally there is the option of a holonomic setup, which is a wheeled base capable of movement in any direction. A holonomic base uses special wheels in an orthogonal configuration which have a passive third degree-of-freedom. This allows the base to move along any vector without having to first rotate. This allows for simpler path planning and obstacle avoidance.
but is complex to control and the wheels suffer in outdoor environments. In most wheeled mobile robotic systems, and all configurations that were considered, the power provided to the wheels is provided by electric motors. Using electric motors has many clear benefits over a petrol powered system. An electric motor control system is easy to implement due to widely available parts and the electric motor control hardware is easy to interface with. There is also the benefit that the motor’s power source can double as the power source of the control hardware and does not need a separate power system to handle the electronics. An electric system is also able to be run indoors without any emissions or harmful effect to people in its surroundings.

1.2.2 Sensors

In order for any robot to perform tasks autonomously it must have some feedback on its own actions as well as current information of its surroundings. Internal sensors provide information about the robot’s own configuration and, for mobile robotics, its position and orientation. This information is primarily used for the control systems but also for the localization system. Common examples of internal sensors for mobile robots would include:

Rotary encoders for measuring angles and rotation. A rotary encoder emits electrical pulses for each minute rotation of the shaft that it is connected to. The number of pulses is a parameter of the encoder itself and can be produced with different amounts of pulses/rotation. A very common type of encoder is a quadrature encoder. A quadrature encoder produces two pulse signals or so called pulse trains which are offset by 90 degrees. With little processing, these signals can reveal not only the amount of rotation but also the direction of rotation and resolve the amount of rotation with a four times increase in resolution. This type of sensor could be used
Figure 1.3: An example of a MEMS accelerometer [6].

for measuring joint angles of a manipulator or more commonly in a wheeled mobile robot the wheel rotation which is useful for pose estimation. The quadrature encoder is the most commonly used sensor but other rotation sensors such as resolvers or tachometers are sometimes used.

Magnetic Angular Rate Gravity (MARG) sensors or an Inertial Measurement Unit (IMU) plus a magnetometer are devices which combine many sensors on one device. An IMU consists of two sensors which measure inertial changes of its orientation; A gyroscope and an accelerometer. These two sensors with the addition of a magnetometer make a MARG sensor. Each of these sensors can and often will be configured to measure in three orthogonal directions enabling each sensor to produce a vector of its measurement in 3D. The function of each device is as follows:

• Accelerometer: An accelerometer is usually a MEMS device which uses a small spring and piezo-electric sensors to measure the force on a proof mass (see
Then the relation (Force = mass times acceleration) is used to determine the acceleration of the device in the three axes. Assuming the sensor is not accelerating relative to the Earth this will provide a gravity vector which points directly to the centre of the Earth. This vector can be used to determine the sensor’s current roll and pitch.

- **Gyroscope**: The gyroscope is also a MEMS device. They work on a similar principle as a spinning gyroscope but instead the internal element vibrates in a plane. This vibration resists rotation perpendicular to the plane of vibration. The element will deflect if the sensor is rotated. This deflection is measured by microscopic moving element capacitors. Gyroscopes produce a signal that represents the angular rate of the sensor in rad/s. This signal can also be used to determine changes in roll, pitch, and yaw.

- **Magnetometer**: The magnetometer measures magnetic flux. These devices also measure the flux in three dimensions which allow the device to resolve the direction of magnetic north in any orientation. Because the Earth’s magnetic field can be approximated as a homogeneous field, the magnetometer can produce an attitude estimate on its own. However it suffers from large amounts of noise and distortions due to local Electro-Magnetic Interference (EMI) and changing magnetic environments.

Each of these sensors come in the form of a small integrated circuit with built in measurement and communication logic. The sensors communicate through a serial interface and are often able to make up to 1,000 samples per second. This serial data can be interpreted by a microcontroller or microprocessor and retransmitted to a host computer for processing.

The sensors in the MARG sensor along with wheel encoders all contribute to the current state of the robot. In fact, each on their own can produce some part of the
total pose. Using a model of the robot’s footprint, encoders alone can produce a whole estimate of the robot’s orientation and position on level ground. The problem with this and with all of the sensors is that there is error that must be accounted for. Wheels will slip on the ground, especially when a differential drive robot turns. This causes error that will build up over time. The issue with the MARG sensor is that most of the sensors produce higher order measurements. For instance, in order to calculate a position estimate from an acceleration measurement the value has to be integrated twice. A static offset error in acceleration will produce a quadratic error in position. Even the smallest offset of one bit in the sensor may relate to 0.0001 \( m/s^2 \) will correspond to a offset of over 1 m in just over two minutes and 100 m in under 24 minutes. In order to successfully use these sensors the different sources of error must be accounted for. This is accomplished though special filtering or sensor fusion in which all sources are sampled and a single pose estimate is produced. This type of processing must favor the strengths of each sensor while removing as much error as possible and is therefore complex. The details of some sensor fusion methods are described in Section 2.4.1.

1.2.3 Navigation and Mapping

In order for a mobile robot to navigate through an unknown environment it must have some way of sensing the world. This is usually accomplished with a directional sensor such as a sonar sensor or grid of sensors. In more complex situations computer vision may be employed or, more commonly, laser range sensors or Light Detection And Ranging (LiDAR) sensors are being used (see Figure 1.4). LiDAR sensors provide an accurate measurement of obstacles in a plane as the sensor rotates. Some LiDAR sensors are capable of sub-degree increments, 50 m range, greater than 180° field of view, and a refresh rate of 50 Hz. Recently LiDAR sensors have become economical
Figure 1.4: An example of a LiDAR sensor [7].

enough for use in mobile robotics and are an attractive choice.

The area scanned by a LiDAR is, however, strictly planar and quite thin. Any obstacle that is above or below the laser scan will not be sensed by the robot. This issue is not often a problem in structured environments where the main obstacles are walls which are consistent across their height, but is an issue in cluttered environments or outdoors.

With a method to sense the surroundings it is possible to start to navigate through the environment. This is accomplished by a software planner and a motion controller system. Almost all meaningful positions for the robot to navigate to would be fixed to the surroundings. This could be a pickup or drop-off point or a point where a manipulation task is needed. However, without processing, the planner has no reference of its current location to the surroundings. To accomplish this task is called localization. Localization is accomplished by using on-board sensors such as the ones discussed in Section 1.2.2 and LiDAR. In this state, using LiDAR, the planner would
only be able to plan as far as it can see but would be able to move the robot towards the destination. A better planner would have some idea of the terrain ahead in the form of a map. If the map has already been made, the planner can make a better global plan and deal with unknown obstacles as they arise. An additional benefit to using a map is that the LiDAR can be used to help with localization by matching its scans to the provided map. This way it can correct any errors incurred by the on-board sensors. If no map is available, which may be the case for an ARMR, it is possible to make a map. This process is called Simultaneous Localization And Mapping (SLAM). This is accomplished using the same sensors as before with a probabilistic algorithm which generates a map using LiDAR sensor information and at the same time estimates the robot’s position on the map. This is an inherently difficult problem that can be solved with different methods, some of which are discussed in Section 2.4.4.

In order for an ARMR to produce a radiation estimate it will need to take measurements over the entire area of interest. This is accomplished with autonomous exploration. Using the localization and navigation functions described above, the autonomous exploration algorithm must produce a route which guides the robot to each point in the exploration area. An intelligent algorithm will be aware of the sensors available and the purpose of exploration. For an ARMR, an intelligent algorithm will be aware of the current radiation estimate and instruct the robot to take measurements in a location which will improve the estimate and reduce error. A survey of algorithms and the applications for an ARMR is further discussed in Section 2.1.1.

1.3 Problem Statement and Requirements

In the event that there may be a radiation source in an unknown location, the traditional technique is to have a human explore the area with a handheld detector,
putting them in danger unnecessarily. The focus of this thesis is the development and testing of an autonomous robot for the purpose of aiding personnel in the localization and mapping of radiation sources, recording the strength of the radiation sources, and doing so without endangering human health. In order to be an effective tool, the Autonomous Radiation Mapping Robot must utilize the tools and techniques described in Sections 1.1 (Radiation Mapping) and 1.2 (Mobile Robotics) as well as a novel integration strategy and mapping algorithm.

The focus of this thesis is on the development and implementation and integration of the software systems of a proof of concept prototype for an Autonomous Radiation Mapping Robot, including all subsystems necessary for a fully functioning system. McDougall et al. have presented a method for the implementation of a radiation mapping system for unknown areas and with sparse data [8–10]. Many of the requirements are similar: the robot must be able to traverse the environment, the robot must produce a map of the environment, and the robot must produce a radiation map. The research conducted by McDougall et al. was considered and built upon, but there are many different requirements for this research. The individual requirements for this thesis and the differences from the previous research are listed below:

- The ARMR must produce a source localization map of one or many sources which is accurate enough to be used as a guide for quick removal or disposal with the minimum amount of exposure.
- The ARMR may take as many readings as necessary to accurately predict the location of each radiation source. The previous research required the system to produce radiation maps with measurements outside of the vicinity of the radiation sources.
- The ARMR must be fully autonomous. The robot must navigate, explore and take measurements as needed without human intervention. In the previous
research the measurement locations as well as navigation waypoints were performed by a skilled operator.

- The ARMR must be able to perform its task in both an indoor and outdoor environment. The previous research was limited to an indoor environment.
- The ARMR must have an easy to use human-machine interface complete with access to all necessary functions and feedback, an easy to understand visualisation of the robot and its environment, and overlays for the source localization and SLAM maps as well as a live heat map of the radiation in the room. This will allow any operator to effectively use and understand the output of the system.

The full development, integration strategy, and live testing of the ARMR system is presented in this thesis.

1.4 Summary of Contents

- Chapter 2 presents requisite background information on radiation detectors and their selection, as well as the Robot Operating System (ROS), a software framework on which the software elements will operate.
- Chapter 3 presents a breakdown of each major component of the ARMR as well as the low level hardware and software used on the ARMR.
- Chapter 4 presents the implementation of the navigation, exploration, and localization subsystems along with their integration with each of the robotic systems. Additionally the Graphical User Interface (GUI) and radiation detection systems are outlined in this chapter.
- Chapter 5 presents the tests, results, and discussion for the validation of the ARMR.
Chapter 6 presents the conclusions and recommendations for future work.
Chapter 2

Background and Literature Review

In this chapter the details of radiation detectors and the Robot Operating System (ROS) will be discussed. Section 2.1 presents a literature review. In Section 2.2 different radiation detector types will be analysed and compared. At the end of this section a suitable detector will be selected for use on the ARMR. Section 2.3 describes ROS. Section 2.4 presents an overview of robot localization and mapping. Section 2.5 presents methods for robotic navigation.

2.1 Literature Review

2.1.1 Autonomous Exploration

Exploration is a task that many mobile robots must conduct. Some applications of exploration include coverage robots, such as robotic vacuums or mobile network robots and exploration robots, such as mobile sensor or mapping robots. More recently, au-
tonomous vacuums have become commonplace in some homes. These vacuums must visit each point in a room at least once in order to clean appropriately and, therefore, must implement an exploration algorithm to accomplish this [11]. There are many exploration methods currently available. Some methods are described below and their advantages and disadvantages with respect to this thesis have been considered.

2.1.1.1 Boustrophedon Cellular Decomposition

Choset et al. [12] describes a method for covering an area with obstacles in place, with the least amount of overlap. Choset refers to this method as boustrophedon or the way of the ox. This exploration algorithm will produce reciprocating motion over the entire area after first intelligently splitting the area into subsections based on the obstacles in the area. This method prioritizes simplicity and efficiency in order to cover the area quickly and using the least amount of energy.

2.1.1.2 Potential Field Distribution

A mobile sensor network may need to have overlapping readings or maintain a certain distance to its neighbors in order to maintain communication. Poduri et al. [13] describe a method for distributing many sensing robots. This method aims to maximize the area covered by the robots while staying close enough to continue communication with a set number of other robots. This method uses a charged particle approach where initially the locations of the robots are all centered on a starting location. Then the locations are simulated to repulse one another and obstacles in the vicinity. The repulsion factor is based on maintaining sensor overlap and connectivity to other robots. This method could be extended for use with a single robot by taking the resultant robot locations and connecting them using an algorithm to produce a path with way-points at each proposed location.
2.1.1.3 Entropy Model

Stachniss [14] describes an exploration strategy which uses entropy to describe the environment. This entropy describes the certainty that a cell of a costmap grid is either occupied or not. The robot can reduce entropy of a cell by moving to the cell and measuring it. Stachniss incorporates the sensor model to determine the information change of all cells detected in a measurement. The goal of the algorithm is then to reduce the entropy of the area of interest below a certain threshold. The algorithm uses two mechanisms to determine the location to sense. The first method finds the closest areas which have high entropies. The next method uses statistical methods to determine the areas which could provide the maximal information gain if sensed. The latter technique is key to quickly reducing uncertainty and producing accurate results early, even with incomplete data. This method differs from the earlier methods in that it can operate without a predefined obstacle map which allows it to be more robust to changing environments. While this method was developed to sense obstacles, it could be modified to sense radiation as well.

2.1.1.4 Radiation Mapping With Entropy Modeling

Cortez et al. [15] used an entropy method for radiation mapping. They used a small robotic base with a small CsI radiation detector and a $60 \times 60$ cm flat grid with a small source on it. They attempted to create an efficient way to move across the grid so that the entropy would be reduced below a threshold. The entropy is modeled as the variance of each radiation measurement made in a cell. Cortez et al. compares three methods. In the first method uniform mapping is used where the robot spends a set amount of time in each cell before moving to the next in a boustrophedon way, visiting cells sequentially, and repeating until the threshold is reached. Next Cortez et al. describe two Bayesian techniques. One method which moves in the same way
as uniform, but takes into account the variance of the cell and only moves on once the threshold is met. The second technique records and averages measurements until its variance is lower than the adjacent cells and then moves to a cell with a higher variance using a gradient function to determine the direction. Cortez et al. found that the uniform Bayesian method in fact outperformed the gradient method in terms of completion time. Cortez et al. attributes this to extra travel time. However Cortez et al. notes that similar to Stachniss’ method, the gradient method will produce a radiation map at each time step over the whole area with an increasing confidence. This gives more meaning to a intermediate result allowing more information to be gathered about the area sooner.

2.1.1.5 Frontier Exploration

Yamauchi [16] describes a method of exploring an area autonomously. This method uses the robot’s existing navigation and obstacle avoidance systems to determine areas which should be explored. Starting with the occupancy grid produced by the navigation system, the algorithm identifies areas where free space borders on unknown cells. These areas are identified by selecting cells that the navigation system’s sensor model has identified as being occluded when the obstacle sensor was pointed in that direction. These areas Yamauchi labels as frontiers, which then can be explored. The algorithm then uses a depth-first search to determine which frontier to navigate to first. When no frontiers exist the area has been completely explored. See Figure 2.1 for an example. This method, like Cortez’ gradient method, may not produce an optimal path and may need to retrace an already explored area. This method, unlike the previously mentioned methods, is able to operate in changing environments and areas which are completely unknown at the start of the trial.
2.1.2 Statistical Source Localization

Locating the source of some emission is a task which is required in many different fields. The task is to determine the parameters of a source based on an observation of its output. It is very important to have an accurate model of all the elements involved with sensing the emissions from the source. A model of the sensor, the source, and the environment are necessary to model the emission from a source. Using this model, the task is to solve the inverse problem of solving for the source parameters, such as
location and intensity, from the received signal. This task is made more difficult by
the fact that the measured signal from the source is modified by unknown parameters
of the source and the environment. Chen et al. [17] describe a system for localizing
acoustic sources using a distributed sensor array. Using time of flight, the arrival times
to each sensor, and the locations of the sensors, Chen et al. are able to determine the
location of the source. However, one obstacle they faced is variations in the speed of
sound. This unknown adds a random element to the inverse problem.

In this thesis, the detection of radiation, especially of weak sources, is quite random.
In order to solve these inverse problems and accurately model the system, a statistical
method must be employed. Hykes et al. [18] used a Bayesian inverse method in order
to localize multiple sources of different types in a room using only six measurements.
Hykes et al. relied on accurate models and precise measurement in order to apply
the Bayesian technique. Jarman et al. [19] also used a Bayesian method in order to
localize sources in cargo containers. Jarman et al., however, used a more simplified
radiation model and focused on modeling shielding such as the containers and boxes
within. It was shown in their results that the localization was accurate enough to
locate a single source precisely.

Localizing multiple sources increases the dimensionality of the problem quite signif-
ically. Using techniques as described before become increasingly computationally
demanding with the addition of more sources. Chin et al. [20] describe a method
which uses Bayesian techniques along with a particle filter. Particle filters are adept
at solving highly dimensional and non-linear problems. Chin et al. used their com-
bined approach to both determine the number of sources in a room and localize them.
Their approach used a set of sensors which were distributed along a grid throughout
the room. Particle methods have a downside in that they may not converge on the
global solution and may converge instead at a local minimum. These methods require
tuning in order to perform efficiently and accurately. When tuned correctly they can
be quite efficient and accurate as evidenced by Chin et al.’s results.

Towler et al. [21] describe two methods of localizing a radiation source using an autonomous model helicopter. The source is placed in a $400 \times 400 \ m$ field and the helicopter is meant to fly over the area and localize the radiation source as quickly as possible. The first method uses a recursive Bayesian estimation technique to search a grid for the presence of a source using a simplified radiation model. The second method uses a contour following method with a Hough’s transform to determine the centroids of the contours. Towler et al. were able to demonstrate the feasibility of both methods and were able to show the speed at which the methods complete the task. Towler et al. concluded that both methods suffer from inaccuracies from measurements of radiation as well as the robot’s position but admit the simulated error used may not accurately describe a real scenario.

McDougall et al. [8] employed a different statistical approach to solve the localization problem. McDougall et al. used a Markov-Chain Monte Carlo method which, like a particle filter, is able to determine parameters in highly dimensional space. In their method they used a more accurate Poisson distribution to model the detector response. McDougall et al. showed that the algorithm was able to localize many sources in a relatively small number of iterations. However, the Markov-Chain Monte Carlo method is more computationally demanding compared to a particle method and therefore does take a relatively longer amount of time.

### 2.1.3 Radiation Mapping Robots

Other robots which were designed to map the presence of radiation sources have been demonstrated in the past. Both McDougal et al. [8] and Cortez et al. [15] made fully operational radiation mapping robots.
McDougall et al. [8] has primarily contributed a novel source localization algorithm as described in Section 2.1.2. McDougall et al., in order to demonstrate the algorithm, also built a fully functional robot and used it to take radiation measurements for analysis. The robot ran ROS and was capable of obstacle avoidance and SLAM making it very capable for indoor environments. The robot however had a passive Castor wheel, making it unsuitable for outdoor operation. McDougall et al. manually chose sampling positions based on personal preference and restrictions in the environment not using any methods described in Section 2.1.1. The robot lacked integration and relied on the operator for motion command and data recording. Using this robot however, McDougall et al. were able to demonstrate a fully functional robot on a practical scale.

Cortez et al. [15] primarily contributed a novel exploration and sampling algorithm as described in Section 2.1.1.3. Cortez et al. used small a small robot in a small known environment to demonstrate their entropy method in [15]. These robots would not be equipped to operated in a real-world scale or an unknown environment. Cortez et al. recently developed and expanded on a method for controlling multiple robots and efficiently sharing work between them in order to reduce the time to localize radiation sources [22,23]. The method Cortez et al. employed is very similar to that described in Section 2.1.1.3 and has been dubbed “information surfing”. The method uses a gradient approach to the information gain that each robot will obtain in the next iteration. These methods provide a smart way to move a robotic radiation robot for the most efficient movement possible, but implementing these methods in an unknown environment would be impossible.
2.2 Radiation Detection

In any robotic sensing scenario there must be a transducer of some kind to sense a parameter of the environment and convert this to an electrical signal that a computer can understand. In order to obtain useful data, the detector’s response must be calibrated and known. Details such as the sensors noise as well as sources of error should also be considered when designing a sensor model or performing analysis on the measurements. For a sensor to be useful in a mobile robotic mapping scenario it must be portable and self-contained. The detector should also be omnidirectional and have a quick response to be suited for a mapping robot. Radiation detection and analysis is a quite complex topic of discussion. For this thesis a detector had to be selected which would satisfy the above requirements and perform the required task of making radiation measurements. In order to localize the radiation sources a few assumptions can be made: i) only concerned with gamma radiation. The other types of radiation are either easily shielded or are only emitted during high energy interactions [24]; ii) the ability to differentiate the energy of each interaction for spectral analysis is considered a benefit. Different detector types were studied and compared to the different requirements described above. After considering cost and ease of use the best detector was selected to be used for this thesis. There are two main types of radiation detectors: ionization detectors and solid state detectors. The next sections will describe the strengths and weaknesses of each type of detector.

2.2.1 Ionization Detectors

Ionization detectors rely on incoming radiation to knock electrons off of molecules in an ionization gas, which is in an electric field. The electrons move toward an anode and the charge is measured by circuitry, which is converted to a count rate and recorded by the instrument [24]. The three types of ionization detectors differ
primarily by the amount of voltage applied to the ionization chamber, listed low to high:

**Ionization Chamber** detectors operate under a very low electric field. This means that the charge generated in the chamber is directly proportional to the present radiation energy. This allows this type of detector to measure very high amounts of radiation as there is no dead-time. The dead-time refers to the time needed for the charge to dissipate from the anode and for the voltage potential to reestablish. This limits the maximum interactions per time period for ionization type detectors. The down side to this type of detector is that the charge generated is very small and advanced circuitry is required to minimize noise and get a clear reading [24].

**Proportional Counter** detectors operate under a slightly higher strength electric field. The electrons generated cause avalanches of new electrons. The benefit of this type of detector is that the size of the avalanche and, therefore, the charge sensed by the detector is directly proportional to the energy of the radiation. This allows this type of detector to distinguish between alpha and beta particles. The disadvantages of this detector is an increased dead-time and these detectors tend to be delicate [24].

**Geiger-Muller Tube** detectors operate under very high voltage electric fields. The electrons always generate avalanches and for each ionizing event a large charge is transferred to the anode, see Figure 2.2. The benefit of this is that the circuitry required to monitor the anode can be very simple and therefore cheap. The disadvantages of this sensor is a greatly increased dead-time, meaning this sensor cannot measure high dose rates. Additionally, the energy of the radioactive interaction is lost due to the large amplification in the chamber and therefore cannot characterize the source [24].
2.2.2 Solid-state Detectors

Solid-state detectors can be split into two categories: scintillators and semiconductors. Both types differ from ionization type detectors in that they do not contain ionization gas. Scintillators and semiconductor detectors also differ from each other. Semiconductor type detectors share a similarity with ionization type detectors in that the incoming radiation interacts with the detector to free electrons. Scintillators work on a much different principle, incoming radiation interacts with the scintillator which produces secondary photons. The photons hit a photo multiplier tube which produces the signal the detector records [24].

Scintillation Detectors contain a material that interacts with ionizing radiation. The material gets excited by the ionising radiation moving it to a higher energy state. When the molecules drop back down to their stable energy state they emit a photon. This photon hits a photo multiplier tube which first converts the photons into electrons and then progressively amplifies the charge. The amount of photons produced is proportional to the energy of the gamma rays, so this type of detector can be used to characterize the source of the radiation. The composition of the scintillation material as well as other factors determine the resolution of the energy response as
well as the sensitivity (see Figure 2.3). Additionally, this type of detector is fully housed to block out external photons making it more robust than the ionization type. The drawbacks are higher cost than ionization types and lower resolution compared to semiconductor types [24].

**Semiconductor Detectors** work similarly to ionization type detectors. When ionising radiation hits the semiconductor it produces electron hole pairs which migrate to plates on either side of the semiconductor. This charge is measured and recorded by the detector. The charge generated is closely correlated to the energy of the incoming radiation. This means that this type of detector has a very high degree of accuracy and resolution when readings are made. The main disadvantage for this type of detector is price and the noise generated by the sensor. To mitigate the noise, most applications cool the sensor with liquid nitrogen [24]. This does not lend itself well to a mobile radiation mapping situation. Having to maintain liquid nitrogen levels may be impossible during a long mission.

Figure 2.3: Comparison of different scintillation compositions [25].
A detector had to be selected for use in this thesis. After considering each detector type and comparing each of their strengths and weaknesses in relation to the requirements discussed earlier, a few options stick out. For practical reasons a semiconductor detector could not be used in a mobile robotic environment, due to the cooling requirements. This left either a scintillation or ionization detector. After considering economic and practical concerns a Na(I) scintillation detector was selected. The detector was available for use and did not need to purchased. The Na(I) detector was also fully contained in a metal shell. This design makes the detector more rugged and less susceptible to interference. Its detection area, in contrast to ionization chamber detectors, is nearly omnidirectional. Over all it is well suited for outdoor mobile use. The scintillation detector also has a intrinsic advantage in that it must be used with a computer interface which makes integration into the ARMR a more straight-forward task.

2.3 Robot Operating System (ROS)

Robot Operating System (ROS) (www.ros.org) is a framework for the implementation of robot centric software on Linux. It allows for the creation and interaction of many different modular program nodes. These nodes can communicate or interact through different provided methods for information sharing and issuing commands. There are different services available which are useful for the control and processing of different sensors. There is a transform system for publishing and looking up transforms from different frames as well as a system for displaying the robot, its current configuration and any sensors in a 3D environment.

ROS has a large community of researchers and robot enthusiasts with a strong open source mentality. Many projects have already been developed for ROS which are open source and are freely available to use. This promotes new development and makes
it easier for beginners to enter the field of robotics but also makes it simpler for researchers to test different concepts easily. Open source projects can also be used as a starting point for a new project because of their tolerant licensing. The development of ROS has taken efforts of hundreds of individuals and thousands of man hours of development time. Development of the features utilized from ROS for this thesis from scratch would have been impossible.

It was decided early in the thesis to use ROS. Previous work by McDougall and von Frankenberg et al. [26] was developed with ROS and the benefits were quite clear. It was then decided that any further development for this research would be developed to be run with ROS. When researching existing solutions to each aspect of the ARMR, any algorithm that already had an implementation for ROS was weighted higher than any others to reduce development time. The next sections describe the different components of ROS used and their purpose.

2.3.1 Nodes

ROS nodes are individual executables. They can communicate with other nodes through ROS’s different methods. Each node is independent from other nodes and has its own memory and its own thread. The only way for nodes to interact is through the ROS methods and because these methods can be piped through common computer networks the nodes can be distributed across different computers. This allows for a smaller and less power consuming computer to be on board the robot. Any implementation of an algorithm for the ARMR will be referred to as a node from now on.
2.3.2 Launch Files and Parameters

ROS nodes can be launched in two ways: either in a standalone manor using the tool `rosrun` or by using a launch file with `roslaunch`. Roslaunch can perform a few different tasks, which make developing and running projects simpler. Roslaunch can launch nodes on the current or a remote machine and interact with the parameter server based on the contents of a launch file. A launch file is a xml text file which describes all of the nodes and parameters that it will load when interpreted. A launch file can also make use of conditional statements and the loading of additional files (which can be other launch files) to produce more complex operations and a more logical distribution of information. Organising a project using launch files prevents the need to launch many nodes and set many parameters manually.

Parameters are variables which nodes can load at run time. The parameters may be set with a command line interface or through launch files or even a node. Namespaces and labels uniquely identify parameters to each node. When multiple instances of a node will be run by a launch file each instance must have a unique name. However, the parameters must also have a unique identifier to differentiate the parameters for the two nodes. In the launch file, parameters and nodes can be loaded with a custom namespace to accomplish this task. Often there are many parameters to set for a given node, separating them into other files decluters the launch file and cleaver naming allows for more logical sorting of parameters. The most useful aspect of using launch files to load parameters is that they are loaded at runtime rather than compile time. This allows a developer to test different values for a parameter without having to recompile, possibly a time consuming task.
2.3.3 Communication Methods

ROS implements many different communication methods. Some are invisible to the programmer and are for logging or coordination of communication, the others can be explicitly used by the programmer for communication to other nodes. The most useful form of communication is through topics. Each topic must have a predefined gdsmessage type. Common gdsmessage types such as velocity or position, are predefined and included with ROS. This makes the creation and utilization of existing open-source nodes a much simpler task. A node can expect data in a predefined format and, assuming that the information is provided, the node should operate correctly. This also makes it simpler to try different algorithms which accomplish the same goal using the same data. It is as simple as switching which nodes are run during the test. Topics, Services and Actions are described in the next sub-sections.

2.3.3.1 Topics

Topics are the core method for nodes to communicate. Topics follow a message board style communication system allowing asynchronous one to one and one to many communications. A topic has two predefined attributes which makes it unique. A topic must have a unique name. This allows nodes to differentiate different topics. A topic must also have a predefined message type. A message is a template for the data that will be transmitted over the topic. A message consists of either a variable of basic data type or a set of variables or other messages or any combination of such. The variables must be named for easier use and the message must be defined at compile time. An example of a twist message is in Listing 2.1 and it is part of the geometry_msgs package. This message is meant to contain linear and angular velocity. Each linear and angular velocity is composed of a Vector3 message which also is a member of the geometry_msgs package. The Vector3 message is then composed of
basic data types for $x, y, z$. This message is compiled into programming headers at compile time for use.

Listing 2.1: Twist Message

```markdown
# This expresses velocity in free space broken into its linear and angular parts.
Vector3 linear
Vector3 angular
```

A topic is created when a node creates a publisher and announces its presence to the ROS subsystem. From that point the node can publish filled messages onto the topic at will, agnostic to the presence of any node listening to the broadcast. A node wishing to listen to a topic creates a subscriber and requests that the ROS subsystem notify it of any new messages. There are checks in place during run time to ensure that both the subscribers are publishing the correct message and the subscribers are expecting the correct message. Many nodes can subscribe to the same topic and each will receive an interrupt when a new message is published.

### 2.3.3.2 Services

Services are similar to topics in that they allow for communication between different nodes. A service is different from a topic in that it is used to trigger an action by another node. A service is defined in a similar way to messages except that they have a defined input and return message. A node first advertises the service and other nodes attach to the service. The client node calls the service by sending the filled call message and waits for the response. The service host receives an interrupt similar to a topic subscription and prepares the response. Once finished it sends the result back to the client. This allows more specific information to be passed between nodes only when necessary.
2.3.3.3 Actions

Actions are very similar to services but are more flexible. The way actions are handled is completely up to the programmer and this allows the ability to override actions when a new action is called, allowing the client to preempt a current task, preventing hang ups. This, more direct system, allows for easier bulk processing and process management. Actions also allow for feedback to be sent by the action server to its client. Actions are best used in more complex implementations where a service would not provide sufficient control over communication.

2.3.4 Transform System

The transform system in ROS (named tf) is very powerful. The transform system consists of joints and frames. A transform represents a frame linked to another frame through a joint. The full set of transforms form a transform tree where each frame has exactly one parent and may have many children. Measurements can be made in any frame and transformed into any other frame using the transform systems Application Program Interface (API). A manipulator can be expressed as transforms and its joint angles incorporated automatically through the use of a node called robot_state_publisher. This information is also used in the robot visualization software discussed in Section 2.3.5 for displaying the robot’s current configuration. The transform system is well suited to be used for manipulators but is also indispensable for navigation. The robot’s current position is referenced through the use of a transform. A frame locked to the map or world is connected to a fixed frame of the robot through a transform. This transform is often calculated using the sensors discussed in Section 1.2.2 but an intermediate transform may be provided when using SLAM. This is discussed further in Section 2.4.4.
2.3.5 Robot Visualiser

The robot visualizer node is called `rviz` in ROS. Rviz is a 3D visualization software capable of displaying many different types of messages by subscribing directly to the relevant topics. For example `rviz` is able to display laser data, or a costmap from a navigation node. In order to position the data correctly the transform system is used. Each data source has a frame associated with it and `rviz` will use this as the origin of the data. A robot model may be displayed in `rviz` by running `robot_state_publisher` with a loaded Universal Robot Description Format (URDF). The URDF contains the configuration of the robot including movable joints and links along with visual meshes, which `rviz` is able to display. Rviz is able to display large amounts of information at once and is therefore a very important tool for development. Rviz also allows for interaction with different nodes. For instance, there is a tool which allows for the control of a mobile robot by publishing twist messages in response to mouse movements. It is also possible to send navigation goals to the navigation stack discussed later in Section 4.1.

Rviz allows new display types, dockable panels, and tools to be developed. This makes it possible to create a fully featured user interface in ROS. New display types can be used to display custom messages. A dockable panel can be used to add any additional controls that might be required to control a robot. This is discussed further in Section 4.5.

2.3.6 Debugging Tools

ROS has many debugging tools which make testing easier. There are tools for checking the connections between nodes and the connections in the transform tree. They are `rqt_graph` for topic connections and `rqt_tf_tree`. Two more tools which allow for
testing and development without access to a robot are gazebo and \texttt{ros\_bag}. Gazebo is a full simulator for mobile robotics. It allows for custom worlds and simulates the environment. A custom model of the robot is needed but custom sensors can be created for gazebo. \texttt{ros\_bag} is used with a working robot and is used to record the communication through topics and transforms as well. The recording can then be played back later to be used to test software without having to rerun a full test. A \texttt{ros\_bag} is desirable for testing because real data is captured instead of a virtual source of measurements.

\section{Robot Localization and Mapping}

As previously discussed in Section 1.2 mobile robots need to be able to know their location in order to be able to navigate reliably through an environment. Different sensors must be coordinated together to localize the robot and to produce a map. Each of these sensors have weaknesses and produce noisy signals. Different methods can be used to attempt to eliminate the inaccuracies of each sensor by combining many sensor sources together. Mapping algorithms also use statistical methods to produce an accurate map from noisy data. The details of the different methods are detailed in the next sections.

\subsection{Pose Estimation}

Pose estimation as previously mentioned is the process of determining the robots current location relative to a fixed frame. The sensors most used are MARG sensors and wheel encoders. These sensors are used in a process called dead reckoning. Dead reckoning is the determination of a robots location based on information about its direction, speed and duration of travel \cite{27}. To incrementally determine a ground
robots position and heading using the previous pose, only the distance traveled forward and amount of rotation since the last increment is needed. This is accomplished with a simple model [27].

\[
\xi(k+1) \sim \begin{pmatrix}
x(k) + \delta_d(k) \cos(\theta(k) + \delta_\theta) \\
y(k) + \delta_d(k) \sin(\theta(k) + \delta_\theta) \\
\theta(k) + \delta_\theta
\end{pmatrix}
\]  

(2.1)

In Equation (2.1) the vehicles pose \( (\xi) \) is described as \( x, y, \theta \) in three rows. The new pose is determined from the previous values and \( \delta_\theta \) the change in rotation and \( \delta_d \) the distance traveled forward since the last update. The problem with using accelerometers and gyroscopes for pose estimation, is that they produce only instantaneous measurements and magnetic sensors and wheel encoders suffer from large errors. Accelerometers and gyroscopes produce higher order information about the robots movement. In order to determine the robots speed and direction, the acceleration and angular rate have to be integrated. This poses a significant problem because even the smallest amount of error will also be integrated and contribute to an ever increasing pose error. To determine position the acceleration must be integrated twice causing a constant offset error to become quadratic error in position. This makes dead reckoning using only a gyroscope and accelerometer to be quite inaccurate over even moderately long distances.

In order to improve this pose estimate, either a sensor which is locked to the world frame must be used or multiple different sensors must be combined to increase accuracy. With the addition of wheel encoders its easy to assume that the wheels must be fixed with the world frame and therefore must provide an accurate estimation of position. This, however, is a false assumption because wheels will slip in any but the most ideal situations. This is especially true with differentially steered robots. When turning, the wheels will continuously slip. This leads the wheel encoders to only be
accurate when travelling in a straight line.

A magnetic sensor is the first sensor that actually is fixed to the world frame. It measures the magnetic field passing through it in three dimensions. This can be used to determine its orientation in relation to the earth’s magnetic field. The magnetic sensor can be used like a gyroscope without the need to integrate the signal. This means that its error will not grow over time. This is useful but a magnetic sensor has its own drawbacks as well. The signal from the magnetic sensor is heavily damped and a settling time must be accounted for. This means that it is not useful for quick movements such as turns in place, a maneuver that robots often perform. The other major issue is that the magnetic sensor is affected by the local environment’s magnetic field which can be affected by electronics or any large ferrous objects. The algorithm needed to produce the magnetic sensors orientation, also needs to be calibrated to the current environment and additionally filtered to get an accurate result.

It is clear that each type of sensor has drawbacks. In order to draw on each sensor’s strengths a filtering technique must be implemented. The details of some techniques are detailed in the next section.

2.4.2 Filtering Techniques

Relying on multiple sensors for pose estimation should make the estimate more robust to error. This involves a process of filtering each sensor’s signal and attempting to draw on each sensor’s strengths and block out their erroneous signals.

2.4.2.1 Complimentary Filter

The most simple method is a complimentary filter. Baerveldt et al. [28] designed a filter for attitude estimation for a helicopter. This filter used high and low pass filters
to weight two different sensors: an inclinometer and a gyroscope. The inclinometer, having a large amount of inertia had a low pass filter applied allowing its signal to pass during slow movement. The gyroscope had a high pass filter to filter out its small offset errors.

2.4.2.2 Explicit Complimentary Filter

Euston et al. [29] created a slightly more complicated method called an Explicit Complimentary Filter. This method also determined the attitude of a flying vehicle but used more sensors and a new filter. It had a three axis accelerometer, gyroscope and an airspeed sensor. The filter’s main difference was that it used an error feedback between the gyroscope and the orientation determined from the accelerometer. Using the error the weighting of the accelerometer’s signal was dynamically adjusted based on its residual with the gyroscope and airspeed sensor. The structure of the filter can be seen in Figure ??.

2.4.2.3 Madgwick Filter

Madgwick et al. [30] developed a filter that is similar in nature to a complimentary filter but approaches the problem as an optimization problem. Madgwick et al. decided to use the Gradient Decent Algorithm to determine the best possible solution to the attitude problem. They first formulate a cost function for the current estimate using a reference position and the sensor measurements then attempt to reduce the cost function using a gradient descent method.
2.4.2.4  Extended Kalman Filter

The above methods along with other basic techniques only work with linear systems. The Madgwick et al. [30] method is more robust to nonlinearity but uses a costly optimization function to accomplish this. Different techniques have been developed to deal with the nonlinearities but by far the most used is the EKF.

The EKF is a special case of the Generic Kalman Filter. The Kalman filter is an optimal estimator for the case where the process and measurement noise are zero-mean Gaussian noise [27]. It is very good at combining many noisy signals and producing a state estimate as well as the uncertainty of the state. The basic stages of Kalman filtering is prediction and correction. In the prediction phase the previous estimate and control inputs are propagated through a system model to produce an estimate. In the next phase the estimate is corrected with a new sensor measurement’s residual. The gain for which the residual is weighted (called the Kalman gain) is set so that the current states covariance is minimized. A high confidence measurement will dominate over a low confidence prediction. Concurrent with the state estimate, the state estimate’s covariance is calculated, providing the filter’s confidence for each estimate. Many signals may be used to feed the update phase each weighted by their own variance. For this to be successful all the signals must represent some change in the output state and must be accurately described in the sensor model. The Kalman filter is a great technique to use, however it only works with linear systems. This is where the EKF comes in. The EKF is an application of the Kalman filter where the system model contains nonlinear equations. It is accomplished by taking a local linear approximation around the current state estimate and applying the Kalman filter stages. This filter will operate the same as the General Kalman Filter, however care must be taken because the uncertainty estimate will no longer accurately represent the uncertainty of the system. This is because the probability density functions are
operated on by non-linear functions. The consequence is that the EKF is not an optimal estimator.

In order to circumvent this issue a new way to represent the probability density function had to be developed. Julier et al. [31] developed a method which uses sigma points to represent the probability density function through each step. They called this an Unscented Kalman Filter or UKF. The differences between a UKF and an EKF can be seen in Figure 2.4. In practice however, the EKF works well for pose estimation of a ground robot. Moore et al. [32] developed a ROS node for sensor fusion which implements an EKF and allows many different sensors to contribute to the state estimate.
2.4.3 Mapping

For mobile robotics, using and creating maps is one of the most important tasks to accomplish. A map can be used for localization, path planning and exploration. A map can come in many forms, but all maps attempt to digitally represent the features and obstacles of the environment which the robot’s sensors can measure. This will either be a geometric representation or, more often, a grid map. A grid map is a matrix where each cell represents a physical area in the real world. This small area is assigned a value. The value represents either a binary occupancy flag, or a probability that the cell is occupied.

2.4.3.1 Map Localization

A map used for localization has one main benefit over an EKF in that it can provide global localization. This means that the localization estimate will maintain its accuracy over the range of the map. Determining a robot’s location using a map is inherently a probabilistic problem, sensor readings and motion commands must be compared with the map and the most probable location determined with each time step. Thrun et al. [34] describe some techniques for localization using a map.

2.4.3.2 Grid Mapping

The first method, called grid mapping, uses a Bayes filter on a cell decomposition of the space similar to the map itself. This method relies on a good selection of grid size to balance accuracy with computational demand. This method is not as robust with complex locations and its high computational demand means this method is not well suited to a practical mobile robot.
2.4.3.3 Monte Carlo Localization

The second method Thrun et al. [34] describe is called Monte Carlo Localization (MCL), named for the Monte Carlo algorithm it uses. This method is a particle method where each particle represents a possible location of the robot and the mean of the particle set represents the algorithm’s location estimate. This method takes time for the particles to converge to the actual robot location but is robust to errors. If the robot moves unexpectedly the particles should reconverge with subsequent movement and sensor readings. If the robot has moved further away from the particle set, random particles can be added to search a wider area and help with reconvergence.

2.4.3.4 KLD-Sampling

Fox et al. [35] describe a MCL method with an improved sampling technique. This sampling technique is called KLD-Sampling which is based off of the Kullback-Leibler distance. This technique dynamically changes the number of particles used by MCL to optimally trade computational efficiency with complexity of the represented probability density. This method is quite efficient and is used by the ROS node Adaptive Monte Carlo Localization (AMCL) for localization [36].

2.4.4 SLAM

In most cases using an existing map alone is not enough, a mobile robot must also produce a map. This is especially useful when exploring unknown space. As the area is explored, subsequent traversals will be more efficient. This task, however, produces an interesting problem. In order to produce an accurate map an accurate pose must be known. As discussed in Section 1.2.2 the error associated with all the sensors commonly used in a mobile robot will grow over time without bounds. The
techniques described in Section 2.4.2 will attempt to produce the best possible pose estimate, but since the error for all sensors is increasing the total pose estimate will continue to increase in error. The only way to limit the error of the system is to use sensors which take measurements which are earth referenced. The techniques described in the last section used a map and range measurements of the environment to produce an accurate location estimate. Creating a map is a much more complex task. The robot must use range measurements to both create a map and estimate its position. These methods are called Simultaneous Localization And Mapping (SLAM) methods. Many of the same techniques used for localization in a map can be used for SLAM. However, instead of just estimating the robot’s location, each of the observed landmark’s position must also be part of the state estimate [27]. This causes the solution space to have a very high dimensionality. High dimensional problems lend themselves well to optimization methods. While other methods such as using an EKF for SLAM [27] have been used, more recently particle filters have been used such as the Rao-Blackwellized Particle filter [37]. Grisetti et al. [37] developed a SLAM algorithm using a Rao-Blackwellized particle filter which improved the computational demand and made the algorithm more efficient. The ROS implementation of SLAM uses this algorithm which Grisetti et al. [37] developed. This ROS package is called gmapping and is discussed further in Section 4.1. New development is tackling the issues with the Rao-Blackwellized approach such as handling cumulative errors when travelling in large loops. Kaess et al. [38] have developed a method which attempts to solve this problem, which has been called Incremental Smoothing and Mapping. There, however, is not a practical implementation of this algorithm for ROS which makes it difficult to implement in the context of this thesis.
2.5 Robot Navigation

Robotic navigation is the task of travelling through an environment. This task can be split into path planning and motion execution. Path planning can further be split into global and local path planning. In general, global path planning attempts to produce a general path from the current position to the goal position taking into account the map and the robot’s size and mobility. Similarly, local path planning attempts to produce a motion command which will best match the global plan over a short time period, taking into account current sensor measurements and the robot’s size and mobility. Path planning is an active area of development and specific implementations are usually tailored to the environment, sensors, and vehicle for which it is intended. General solutions for global and local path planning will be discussed here.

2.5.1 Global Path Planning

Global path planning has its theoretical roots in computer science and graph theory. For ground robots which use a 2D occupancy grid as their map, the task is to create the shortest path from the starting grid element, to the finish element, without crossing an occupied cell or passing through a gap which is smaller then the robot’s footprint. The common algorithms used to accomplish this task is the Dijkstra and A* algorithms [39, 40]. Each of these algorithms will find an optimal path by exploring from the starting position, moving towards the goal and testing each path for its cost, and then exploring in the direction of lowest cost. The cost for a ground robot will be the distance travelled. A* is similar to Dijkstra but incorporates a heuristic in order to rank paths when there are multiple equal paths. The A* algorithm should find an optimal solution in fewer steps than the Dijkstra algorithm. These algorithms are based on graph theory and must be adapted to work in real environments. The D* algorithm is an adaptation of A* for use with a cost map [27]. This algorithm incorporates a
cost model that accurately reflects the cost map and has the benefit of incremental replanning. LaValle [41] describes the A* and Dijkstra algorithm as well as many extensions to the path planning problem and their solutions. Recently, there has been research into path planning in highly cluttered environments. Barraquand et al. [42] developed path planning algorithms for robots which are highly dexterous. This could allow a robot to manoeuvre in a constraining environment such as a collapsed building, where a simple planner would not be able to guide the robot efficiently. The ROS navigation suite uses the Dijkstra algorithm which has been modified to work with cost maps [43]. It can, however, have the option to use A* if necessary, however this was not permanently changed due to the lack of significant improvement over Dijkstra’s algorithm. This is due to the fact that the global plan is only intermittently requested and the computational demand is low.

2.5.2 Local Path Planning

The local path planner has a very different task from the global path planner. The local planner is given the task of developing a motion command which is achievable by the motion controller and which does not result in a collision with any obstacles in the path of the robot. The algorithms discussed above could be used to develop the local path plan if modifications were made to restrict the plans to those which are achievable. For example, only produce paths with a minimum radius for the case where a robot cannot turn in place. These algorithms, however, are computationally demanding in a local planning environment where continuous replanning is necessary. The need for computationally efficient algorithms is increased by the fact that obstacles close to the robot may be moving. This demands that the algorithm must either recalculate a path to incorporate new obstacles often, or incorporate an intelligent system which will predict the location of obstacles in the future. Gerkey
et al. [44] developed a local planner for unstructured terrain called trajectory roll out. This method samples the control space for the robot, and simulates the forward kinematics of the robot over a short time period. It then ranks each trajectory based on the map and how much further it brings the robot along the global path. A trajectory is chosen based on rank, and the process continues from there to the global goal. The achievable velocities of the robot are incorporated by including them in the sampling technique. This method is used in the ROS navigation suite as the default local planner but other planners are possible. Another popular ground vehicle local planner which has been developed to be used in ROS is the Dynamic Window Approach (DWA). This method Fox et al. [45] developed is very similar to trajectory roll out. DWA differs from trajectory roll out in that it samples a set of trajectories, not motion commands. The limits of the robot’s motion controller have to be applied after trajectory ranking and can affect the quality of the trajectories chosen. However, DWA samples a smaller dimensional space than trajectory roll out and can prove to be more computationally efficient.
Chapter 3

System Overview

3.1 Functional Overview

The ARMR can be broken down into different functional parts. Figure 3.1 shows the flow of information through each functional component of the ARMR. Each sensor is connected to a ROS connected computer. The sensor signals are processed by input processing nodes to convert the low level signals to ROS compatible messages to be used by other nodes. This is accomplished with existing ROS nodes. The ARMR can be broken down into five main parts:

- The **navigation** subsystem is responsible for moving the ARMR through the environment. It takes input from the sensors, exploration subsystem, and the control node. It sends signals to the exploration subsystem and the control node. For the full implementation details see Section 4.1.

- The **exploration** subsystem is responsible for generating plans for exploring the area of interest and selecting points to take measurements. It receives signals from the input nodes, navigation subsystem, and the GUI. Its output is to the navigation system. For the full implementation details see Section 4.2.
• The **URSA II** node is responsible for reading the signal from and controlling the URSA Radiation Alert DAC. It produces a custom ROS message which is used by the source localization subsystem and the control node. For the full implementation details see Section 4.3.

• The **source localization** subsystem determines the parameters of the radiation sources in the area of interest. It receives commands from the control node and radiation measurements from the URSAII node. For the full implementation details see Section 4.4.

• The **GUI** is responsible for displaying the output from each system and receiving the input from the user. For the full implementation details see Section 4.5.

• The **control node** also had to be developed for the ARMR. It facilitates coordination between each node. It also produces the live intensity map. For the full implementation of the intensity map see Section 4.5.2.
3.2 Hardware Overview

The ARMR physically consists of a robotic base and external sensors. The robotic base is an Unmanned Ground Vehicle (UGV) manufactured by Clearpath Robotics called the Jackal (see Figure 3.2). On the Jackal there is attached a LiDAR and sodium iodide (NaI) radiation detector.

3.2.1 Robotic Base

The Clearpath Robotics Jackal is a robust and compact mobile ground robot. The robot consists of a rectangular body with four wheels around the perimeter. Inside the body is an electric battery power system and a full computer with WiFi and Bluetooth. The computer allows the Jackal to control its motors, process its sensors, and perform some or all of the higher level tasks. The computer’s connectivity allows it to communicate with other computers running ROS through WiFi or allows a direct
method for driving the robot through Bluetooth. Also inside are two electric motors which provide movement through belts which connect to the wheels on each side. The two wheels on each side are connected together and cannot be turned independently. The wheels are also on fixed axles which only allows the Jackal to be differentially driven. The motors are also equipped with optical encoders. The encoders provide the rotation of the motors and through a fixed gear ratio they provide the rotation of the wheels. This provides the odometry for the robot. To control the motors and to provide power to the computer is a control board. On the control board there is also a MARG sensor. The MARG sensor’s output is published to ROS through the on board computer.

3.2.2 External Sensors

The Jackal has two mounting locations on the top of its body. The front position holds a LiDAR and on top of it a GPS sensor. The rear mounting location was used to mount the radiation detector.

3.2.2.1 LiDAR

The LiDAR mounted on the front of the Jackal is a LMS111 model manufactured by SICK AG. The LMS111 has the following specifications [46]:

- Suitable for outdoor use.
- A 270° field of view.
- A Scanning frequency of 50 Hz.
- A working range of 0.5 m to 20 m.
- An angular resolution of 0.25°.
The LiDAR is powered by the Jackal and communicates with the internal computer through an Ethernet connection.

For more information on the GPS hardware and implementation see Section 4.1.2.4.

### 3.2.2.2 Radiation Detector

The rear mounting location was available to mount the radiation detector. The radiation detector is a Rexon 2” × 2” NaI type with a photomultiplier tube permanently affixed. The photomultiplier tube is connected to a computer interface through a cable. The computer interface provides the high voltage the photomultiplier tube needs, as well as processes the pulses from the photomultiplier tube. The readings are sent to the internal computer through a serial port which the interface sends to a software node running on the computer. The software node is discussed in more detail in Section 4.4.

In order to mount the radiation detector a bracket was fabricated to hold the detector and match the hole pattern on the Jackal. The bracket was designed to hold the detector firmly and to not obstruct the NaI crystal. A picture of the bracket can be seen in Figure 3.3.

### 3.3 Input Processing Nodes

In order to publish the data from the internal sensors and perform pose estimation there are a few ROS nodes which run on the Jackal’s internal computer.
3.3.1 Jackal Node

The jackal has its own ROS node called `jackal_node`, which publishes the internal sensor’s data and subscribes to motion commands. The node publishes the raw IMU data from the sensors to the topic `/imu_data_raw`. A node called `imu_filter_madgwick` executes the filter developed by Madgwick discussed in Section 2.4.2 [47] on the data. The resultant filtered IMU data is published to `/imu_data`.

Jackal’s node also publishes the robot’s pose based on the wheel encoders. This is accomplished using inverse kinematics for differential drive robots. This is a combination of a simple equation which calculates the velocity of the robot based on the wheelbase and the wheel diameter, then uses Equation (2.1) to determine the robot’s pose. The pose is published to `/jackal_velocity_controller/odom` The node also performs the forward kinematics to determine motor velocity from desired velocity subscribed from `/jackal_velocity_controller/cmd_vel`. 
3.3.2 Robot State Publisher

The robot’s frames and model are generated by the `robot_state_publisher` node. The Jackal has a URDF which includes the transforms for each of the mounting locations, such as the front LiDAR mounting point or the back mounting point which will be used for the radiation detector. The `robot_state_publisher` listens to a joint state message topic `/joint_states`, to update any movable joints or transforms. On the Jackal the only movable transforms are the wheels which have their own transform in the URDF.

3.3.3 LMS1XX

The LiDAR interfaces to ROS through a node named `LMS1xx`. This node is designed to work with any of the 100 series LiDARs made by SICK AG. The node communicates with the LiDAR over Ethernet and fills a laser scan message which is published to the `/scan` topic [48].

3.3.4 GPS Processing

The built in GPS unit, like many others, uses a standardized communication protocol called NMEA. NMEA stands for National Marine Electronics Association, an organization which has standardized a communication protocol for marine sensors and control systems. Many early GPS systems were designed to be compatible with marine systems, so they adopted the same protocol. The protocol uses a sentence structure which consists of a start delimiter, followed by a comma-separated sequence of fields, followed by the character * (ASCII 42), the checksum, and an end-of-line marker [49]. The ROS node `nmea_topic_driver` receives these sentences and decodes them. The node then fills a `sensor_msgs/NavSatFix` message and publishes it on the
In order for the GPS data to be used in the EKF, the latitude and longitude coordinates have to be transformed to a coordinate system which can compare to the other odometry sources and to the desired output of the EKF. The `navsat_transform_node` transforms the latitude and longitude into the UTM or Universal Transverse Mercator coordinate system. UTM is a two dimensional Cartesian coordinate system where any position on Earth can be referenced with a grid number and a measurement in meters north and east of the southwest corner of the grid. Since the UTM coordinate system is referenced to the Earth in meters, the Navsat node can create a globally fixed `nav_msgs/Odometry` message which it publishes to `/odometry/gps`. This message includes the covariance estimated from the GPS’s horizontal dilution of precision signal.

For more information on the testing and implementation of the GPS system see Section 4.1.2.3.

### 3.3.5 Robot Localization Node

The node responsible for sensor fusion is the `robot_localization` node. It is a practical implementation of a EKF for the purpose of pose estimation. The node may take an arbitrary number of sensors using standard ROS message types as long as the frame in which the sensor is published has a path through the transform tree to the “`base_link`” frame. The node then combines the sensor measurements to produce an overall position estimate using an EKF [50]. The estimate is then published as a transform from a static frame to a frame fixed to the robot.

Each sensor must be defined in the launch file parameters in order for them to be used. Each sensor definition defines usable axes, of rotation and translation as well as
their derivatives. Listing 3.1 shows the parameters used to configure one sensor. Each true or false statement tells the node if the corresponding axis/derivative combination should be considered. The first row describes the $x$, $y$, and $z$ position measurements. The second row describes the roll, pitch, and yaw measurements. The next two rows follow the same order but correspond to the velocity equivalents. The last row corresponds to the $x$, $y$, and $z$ acceleration measurements. For example, Listing 3.1 would accurately describe the useful components from a odometry estimate created from encoder measurements. The $x$ and $y$ positions will be used as well as the yaw. The forward velocity and yaw rate will also be used. These parameters can be changed based on which sensors will produce a good estimate. For instance, if the yaw measurement from the encoders is not trust worthy and another sensor such as an IMU is available for yaw measurement the corresponding parameter can be set to false for the encoder input.

**Listing 3.1: Sensor Configuration**

```
<rosparam param="odom0_config">[true, true, false,
false, false, true,
true, false, false,
false, false, true,
false, false, false]</rosparam>
```

Many of the EKF parameters are hidden to make using the package easier, however, two parameters are exposed for tuning the EKF. The first parameter is the initial pose covariance. Tuning this parameter improves the time the filter takes to converge. The initial pose covariance parameter is difficult to tune so that the convergence time is less and not more and so this parameter has not been set on the ARMR. The result is that the filter takes a marginally longer amount of time to converge than it possibly could achieve. The second parameter is the process noise covariance. This is the uncertainty associated with the prediction stage of the EKF. This parameter is also
difficult to tune. Different values have been tested with the ARMR with varying results. The decision was made to remove this parameter and use defaults in order to simplify the tuning of other parameters.

The Robot Localization node has a unique feature which makes using Earth referenced sensors easier. The issue with most Earth referenced sensors, such as GPS, is that they update relatively slowly and, while the error is bounded, they can have a quite high absolute error. This can cause a significant amount of drift in the localization estimate. A common solution is to use two Robot Localization nodes, which is the case on ARMR. The first node combines the sensor data of the encoders and MARG sensors. This provides a stable and fast state estimate, but has error which grows unbounded. It provides the transform from \texttt{odom} to \texttt{base_link}. The second instance of Robot Localization combines all sensor sources including the GPS. This will produce an unstable estimate but with bounded error. To complete the transform tree, the node will calculate the transform from a static frame to a frame on the robot subtracting the transform provided by first instance of Robot Localization. The published transform, called \texttt{gps} on the ARMR, will point to \texttt{odom}, the fixed frame from the first instance. This allows different nodes to either use a globally fixed frame or a more stable, locally accurate, fixed frame. This is primarily used for navigation purposes and is discussed more in Section 4.1.

### 3.3.6 Command Velocity Multiplexer

Nodes which would like to send motion commands to the robotic base can publish messages to the \texttt{/jackal\_velocity\_controller/cmd\_vel} topic. The Jackal will use these commanded velocities to control the motor controllers and achieve the desired velocities. However, when multiple nodes publish different velocities, the robotic base will try to achieve each velocity in the order in which it receives them. This can cause
the base to jitter as it toggles between different velocities. This will also reduce the accuracy of the encoder values as this can cause slipping.

It was decided to run a node called `yocs_cmd_vel_mux` [51]. This node creates and multiplexes many velocity command topics to one topic. In this case the output topic was the original Jackal control topic. The messages from these topics are given a priority and time out time. The topics created were in order of priority: `/cmd_vel/joystick`, `/cmd_vel/remote`, `/cmd_vel/maskable`, `/cmd_vel/navigation`, and `/cmd_vel`. The node will pass on the highest priority message to the output as long as the timeout is not reached on that topic at which time the next highest message will be relayed. The highest two topics joystick and remote are in place to receive the messages from both the bluetooth joystick directly connected to the robotic base and the thumb stick control in the GUI connected over WiFi. These two velocity sources act as emergency overrides in the case of a malfunction of the autonomous navigation subsystem or a network disconnect. The next two topics are used by the autonomous controls and are discussed further in Sections 4.1 and 4.2. The lowest priority topic `/cmd_vel` was put in place to catch the output of any node which did not or could not be modified to work with this node, and is publishing to the default topic.

### 3.3.7 Bluetooth Node

A node was needed to produce velocity commands from a Bluetooth joystick. The joystick serves two purposes: as a means to control the robot without a computer, and to serve as an emergency override to stop the robot in the case of a malfunction. The Jackal base will not move when it is disconnected from a network. However, a means to control the robot in the case of a crash on the base station computer is a necessity. The node which connects to the joystick is simply called `joy` [52] and the node which produces the motion commands is called `teleop_twist_joy` [53]. The
joy node connects to the joystick and publishes a sensor_msgs/Joy message. The
teleop_twist_joy subscribes to this topic and publishes a geometry_msgs/Twist
message which it publishes to the /cmd_vel/joystick topic.

The teleop_twist_joy node includes a dead-man switch so that no messages are
published while the button is not pressed. This could be used in a reverse fashion to
stop the robot by pulling the dead-mans switch with no input on the thumb stick to
stop the robot while either the base station computer is shut off or the robotic base
is shut off.
Chapter 4

Navigation, Exploration, and Source Localization

Navigation, exploration, and source localization subsystems are the core of the ARMR system. The navigation subsystem moves the ARMR, produces a map of the area, and localizes the robot in space. The exploration subsystem gives instructions to the navigation subsystem in order to lead the ARMR throughout the area to be investigated in order to completely cover the area. The source localization subsystem uses the localized position of the robot from the navigation subsystem along with the measurements taken at the locations from the exploration subsystem to calculate the position and intensities of the radiation sources. In order to measure the radiation intensity at each position, the radiation detection subsystem interfaces with a radiation detector and produces a ROS message. To interface with the different systems a GUI displays the outputs from the different subsystems and coordinates input from the user to each of the subsystems.
4.1 Navigation and Localization Subsystem

Navigation and localization are vital functions of a mobile robot. As previously discussed, navigation is the process of moving through an environment. In order to take radiation measurements the ARMR needs to be moved throughout the environment and around obstacles. At each waypoint the robot’s position needs to be recorded for the radiation source localization algorithm. This task needs to be as accurate as possible in order to ensure the accuracy of the source localization algorithm.

4.1.1 Navigation

Figure 4.1: An overview of the interactions of the “move_base” package [54].

Navigation is accomplished by the node move_base. This node is responsible for coordinating the local and global costmaps and path planners. Figure 4.1 shows the internal and external interactions that move_base utilizes. On the left side are inputs for the robot’s current position, provided by a localization software node or nodes. On the right are inputs for the different obstacle sensing nodes and the map. On the top and bottom of Figure 4.1 are the inputs and outputs of move_base. The move_base inputs (top of figure) provides an interface for movement waypoints. The move_base output
(bottom) is an output interface for motion commands which must be interpreted by the robotic base. In ARMR’s case they are executed by the `jackal_node`.

The waypoint goals can be published to `move_base` in any transform frame, since `move_base` will convert the waypoint. The global planner develops the global plan and the local planner attempts to follow the plan. As discussed in Section 2.5, each planner needs to be tailored to the robot’s specific configuration and capabilities. Clearpath Robotics provides a configuration they have tested for use with the navigation stack. This is a good starting point but it must be modified to match the needs of an indoor close quarters scenario and for an outdoor open space scenario. The specific configurations are described in Sections 4.1.1.1 and 4.1.1.2.

### 4.1.1.1 Indoor Navigation

In the indoor configuration the ARMR uses LiDAR as a sensor source and `gmapping` as a map source. The initial configuration performed fairly well during testing and only needed some slight adjustments in order to prevent the robot from getting stuck and move safer. The travel speed was reduced to $30 \text{ cm s}^{-1}$ from $50 \text{ cm s}^{-1}$, this helped match to robots speed to its environment. The robot would travel very near to obstacles and cut corners close. To solve this problem the inflation radius was increased from $25 \text{ cm}$ to $45 \text{ cm}$. This increased the cost of traveling near to obstacles. However, the cost of traveling near obstacles is a gradient. The local planner would still plan to cut corners to reduce travel distance. The local planer’s trajectory scoring was adjusted to allow planed trajectories to stray from the global plan further. This allowed the local path to take corners wider and safer. The `gmapping` configuration was kept unchanged.

The performance of the internal computer was a concern due to the fact that the navigation and SLAM was moved from being run on the base station computer to the
on-board computer. The individual control loop times of each navigation component were relaxed in order to prevent the on-board computer from locking up.

4.1.1.2 Outdoor Navigation

The outdoor configuration differs from the indoor configuration due to the differences in terrain and situation. The outdoor terrain is relatively empty and large. The first change made was to decrease the resolution of the map from 2 cm to 5 cm in order to improve performance with large maps. Depending on the quality of the localization system and the range of the obstacles, the quality of the map generated by gmapping will suffer and the quality of the map localization will suffer greatly. In order to continue to use gmapping and not have its localization publish incorrect updates, the scan matching minimum score parameter was set to a higher value. This way gmapping could be run in sparse environments. It was later decided to not run gmapping during testing due to the nearly empty test environment and the unreliable output.

The move_base parameters were changed due to the different localization strategy for outdoor navigation. The outdoor configuration utilizes a GPS and a compass for global localization. Both of these sensors introduce a source of error into the position estimate. Since goals are published in the global frame, these errors will show as error in the robot’s position relative to the goal. In order to prevent oscillation, the goal tolerance was loosened. The rotation tolerance was also changed from 0.157 rad to 0.628 rad. Due to the fact that the compass has a relatively long settling time, this greatly improved performance.
4.1.2 Localization

Localization is a core component of a mobile robot. This function is separate from navigation but is required for the navigation subsystem to work. Other functions of the robot also require an accurate view of the robot’s current location. The exploration algorithm and source localization algorithm both need to know the robot’s estimated location.

4.1.2.1 Robot Localization Node

The ROS Robot localization node is a node designed to perform EKF localization as is discussed in Section 2.4.2. This node is purpose built for localizing a robot using different sensor sources. The localization node allows the use of any number of sensors using the standard ROS messages such as nav_msgs/Odometry,
geometry_msgs/Twist, sensor_msgs/Imu, and geometry_msgs/Pose. This allows the use of multiple sensor sources. On the ARMR the wheel odometry and MARG sensors are combined in a robot_localization_node. This node is part of the default start up script on the on-board computer in the Jackal and this instantiation is called ekf_localization. The robot_localization_node outputs its state estimate on two interfaces. The transform from the published “odom” frame to the robot’s local frame “base_link” is published. The estimated position is also published in an odom message. Figure 4.2 shows the different interfaces of the two robot_localization_node used on the ARMR. The second robot_localization_node is called global_localization and is discussed later.

4.1.2.2 Multiple Robot Localization Nodes

The ekf_localization node can be used “as is” for indoor localization. For outdoor localization the GPS and compass sensors should be used in order to produce an Earth fixed location estimate. The sensors used in the ekf_localization node produce a relative location to the initial start location. Using the compass and GPS, the EKF localization will be able to produce a location estimate which is fixed to the Earth. The navsat_transform_node is used first to convert the latitude and longitude to UTM coordinates. The node then produces a nav_msgs/Odometry message referenced to a UTM datum which is either provided or is set as the current location.

Incorporating all sensors into one robot_localization_node is desirable but can cause some issue with navigation and mapping. The GPS sensor can introduce sudden jumps as it gains and loses signal lock with different satellites. On the other hand, the navigation and mapping system require a continuous position estimate. The desired configuration then would be to have one transform representing the combination of all sensor sources and one transform which is continuous and combines all continuous
sensor sources. The Earth fixed frame will be used for publishing movement goals and the continuous frame will be used for navigation. With `robot_localization_node` this is possible.

In addition to `ekf_localization` another instance of `robot_localization_node` called `global_localization` was created. This node combines all sensor sources including the GPS. This will produce the Earth fixed frame named “gps”. Having two `robot_localization_node`s poses a problem however. Both nodes produce a transform from their fixed frame to the robot’s local frame. A frame with two parent frames is not possible in the ROS TF framework. The `global_localization` node needs to provide a transform from “gps” to “odom” instead of to “base_link”. The transform from “gps” to “odom” is just the transform from “gps” to “base_link” less the transform from “odom” to “base_link”. The `robot_localization_node` will compute this, given the correct parameters as input.

The `global_localization` node is launched along with the `navsat_transform_node` when the outdoor configuration is selected in the GUI.

### 4.1.2.3 GPS Tests

In order to localize the robot and achieve high location accuracy in outdoor tests, the GPS was relied upon for accurate location estimates. When the robot traverses open areas where the SLAM algorithm has no obstacles to use as landmarks, the robot will have to rely on the GPS and on board odometry to produce a location estimate. However, during preliminary tests in the Polonsky Commons at UOIT it was noticed that the GPS accuracy was greatly affected by the buildings surrounding the commons. The surrounding buildings block GPS signals from satellites near the horizon, which contribute greatly to the accuracy of the position estimate. The buildings also reflect other GPS signals which introduce multi-path errors into the estimate. After
conducting two tests gathering position estimates from a stationary GPS receiver stationed within the Polonsky Commons and then an open parking lot on campus, the difference in accuracy can be seen. The tests consisted of starting the GPS receiver, letting the receiver acquire satellites and letting the estimated accuracy settle, then recording the position estimates over the course of 5 minutes and plotting them on a graph.

In the first test the position wandered around in an area about 4 $m$ across. The results can be seen in Figure 4.3. While 4 $m$ is acceptable accuracy for most civilian needs, the accuracy of the results of the PSO algorithm and the heatmap will be affected by this drift. It is unlikely, due to the high variability of the cost function, and its bad response to erroneous data, that the PSO will converge with position data that is as inaccurate as this.

![Figure 4.3: Position estimates of a stationary GPS Sensor stationed in the Polonsky Commons, UOIT.](image)

Figure 4.3: Position estimates of a stationary GPS Sensor stationed in the Polonsky Commons, UOIT.
At the second location the results were quite different. The accuracy has improved to about 1 m as seen in Figure 4.4. This may be accurate enough to produce usable data for the PSO algorithm when combined with the position estimate developed from the on board odometry and the position estimate from the SLAM algorithm.

Figure 4.4: Position estimates of a stationary GPS Sensor stationed in a parking lot at UOIT.

However, it was decided to invest in a more accurate positioning system to increase reliability and accuracy especially when traversing open terrain. To accomplish this, an RTK GPS system was selected be used. An RTK system improves on a standard GPS system by leveraging two different techniques. The first technique involves analysing the carrier frequency of the GPS signals and not just the transmitted data. Since the carrier frequency is much higher than the data that it carries, the system can more accurately identify the apparent distance to each satellite. The second technique is differential GPS. Differential GPS is the process of applying double differencing techniques using two GPS receivers to effectively eliminate distortions caused in the
atmosphere. The combination of both techniques is RTK GPS [55].

4.1.2.4 RTK GPS

The RTK GPS system selected for the ARMR is a kit developed by Igor Vereninov. It consists of two inexpensive GPS units with a single board computer paired with each GPS to do the RTK processing. The RTK processing is accomplished with an open source software package known as RTKLIB [56]. This software was developed to perform any of the RTK GPS functions that a proprietary system could, provided that the user has compatible hardware. The system is capable of outputting NMEA messages which made it possible to interface RTKLIB with the existing nodes in the system.

The RTK system was tested in the same parking lot environment as before. A sensor was placed on the Jackal and the base station was set at a surveyed point. The Jackal was driven to a point in the parking lot and the $x$ position of the robot’s position was recorded. The conditions on the day of the test were not ideal with clouds and some rain, however, the results were much more promising. The max deflection over ten minutes was $\sim 20 \text{ cm}$, a much better result. The accuracy of the RTK system was deemed adequate for integration with the source localization system.

4.1.2.5 GPS Processing Tests

The navsat_transform_node was used to convert the NMEA messages from the RTK system to UTM. The navsat_transform_node requires a few parameters to be set. The orientation of the magnetic sensor’s reported direction with respect to true north is needed. This is used to align the robot with the UTM grid and corrects the offset between the GPS antenna and the robot’s fixed frame. The navsat_transform_node also allows the specification of a fixed datum. If a datum is not specified then the
Figure 4.5: Position estimates of a stationary RTK Sensor stationed in a parking lot. Here the $y$ axis is the $x$ position of the detector and the $x$ axis is time in seconds.

position that the robot is in when the node starts is used as a datum. To facilitate comparing different runs of different tests and previously surveyed points a manually set datum needs to be set.

Using a datum, the output odometry will be relative to the datum. Through testing, an offset was discovered between the measured locations and the reported odometry. The datum supplied to \texttt{navsat\_transform\_node} was rounded up to the fourth decimal place. The fourth decimal place of a single degree of latitude or longitude is around 11 meters. With rounding, the datum can be moved by around ±5 meters in each direction. Fortunately the datum which is actually used is reported by the \texttt{navsat\_transform\_node}. This datum can be used to compare runs from different tests and surveyed locations.
4.2 Exploration Subsystem

In order for the ARMR to navigate through the environment, an exploration system needed to be developed. In a known environment navigating is a more simple task. The coverage of the radiation sensors can be evaluated based on the known configuration of the borders and obstacles. In unknown environments, the robot must move throughout the entire area based on limited information.

4.2.1 Indoor Exploration

In an indoor environment, the obstacles in the room allows SLAM navigation to be used and a more advanced exploration algorithm. For this task the frontier exploration node developed by Yamauchi (discussed in Section 2.1.1.5) was used. The ARMR is equipped with the sensors needed to perform frontier exploration. Frontier exploration also requires a map input to identify traverseable areas and to produce a persistent map. This was accomplished with the gmapping node.

The exploration node inputs are an exploration polygon and sensor and map sources, and from that produces way-points based on frontiers. The frontiers, as discussed earlier, are defined as the threshold between known empty space and unknown space. The other thresholds (empty or unknown to full) are barriers. The algorithm searches for these thresholds within the polygon and implements a greedy algorithm in choosing the next frontier to travel to. The next navigation way-point will be the closest frontier. When no more frontiers exist, exploration is finished. The exploration costmap is updated by the view of the laser and the obstacles obtained from gmapping. The empty space from gmapping can be ignored so that the exploration costmap is relatively isolated. Then, by adjusting the laser’s assumed range (regardless of the actual range), the amount of space declared as empty can be controlled. This will not affect
gmapping, only the exploration costmap. Doing this will effectively control the spacing of consecutive frontiers and therefore the spacing of the movement way-points. The effect is similar to a potential field distribution of way-points but also taking into account obstacles which are not initially known. The control node will use the travel way-points to instruct the source localization subsystem to take a measurement as discussed in Section 4.2.3. By changing the LiDAR range, different sample spacings can be achieved.

The algorithm may not be efficient in exploring areas quickly. Due to the greedy algorithm, the exploration algorithm may choose to travel in one direction for quite some time before exhausting the frontiers and then back traveling to the next closest frontier. Another problem that can arise is the navigation subsystem may fail to move the robot to the frontier. If this happens, the algorithm ends. This is necessary because a race condition can arise if a frontier is unreachable. The closest frontier will be the unreachable one and exploration would halt.

4.2.2 Outdoor Exploration

In situations where map navigation would be difficult, a different exploration system had to be developed. This alternate exploration algorithm is controlled by the user in a very similar way to the frontier exploration method to maintain its ease of use but takes a different approach to exploring the area.

The outdoor exploration algorithm takes as input a four-sided exploration polygon and produces a plan for exploration in the form of waypoints. These waypoints are used as instructions for where to take measurements and are handled by the navigation system. The generated plan needs to cover the entirety of the area of interest and do so evenly. The proposed method uses a variable grid to accomplish this.
The operator first decides on an appropriate row spacing and goal spacing. This could be based on the environment searched, the size of the area searched, or the desired accuracy of the source positions. Then the operator draws the search polygon one corner at a time until the polygon is closed. The operator may use the visual one meter grid or a loaded map as a guide for drawing the map.

The algorithm then begins decomposing the space into different points with the spacings previously entered. The first, second, and final edges formed by the polygon are then used to create unit vectors along the bottom, right, and left edges, respectively. Then, using the vectors, a reduced search space is created which is smaller by a preset padding value. This padding value guaranties that the robot can reach the outer most points generated by the algorithm should the operator choose to use a wall as a reference when drawing the bounding polygon. The vectors are then scaled to match the spacing values specified earlier. The bottom vector is scaled by the goal spacing factor and the two side vectors are scaled by the row spacing factor. Figure 4.6a shows the initial state of the exploration algorithm. Here the goal spacing is larger than the row spacing for demonstration. Also note that this exploration polygon is rectangular, however four-sided polygons with sides which are not parallel will also work.
Starting in the bottom left of the padded polygon, the algorithm moves a point back and forth, moving up with each pass. This is accomplished by adding or subtracting the bottom vector. If the next point generated would fall outside the padded polygon it will replace it with a point on the edge of the polygon. Then, using the appropriate side vector, the point will be moved up the side. The point then moves back along a parallel track by once again using the bottom vector. When the point escapes the top edge, the algorithm adds one last pass along the top edge using the bottom vector once again.

Figure 4.6b shows an example of how the algorithm would finish given dimensions which are not multiples of the row or goal spacing. In this figure, a waypoint would be saved at the tip of each vector. It can be seen in the figure how the algorithm may not have an ideal distribution of points near the edges of the search area. Conversely, the algorithm has successfully covered the area completely. The points generated are now ready to be used to move the robot.

The exploration algorithm then proceeds the same as in Section 4.2.1, except that each new way-point has already been generated. The navigation system will take care of moving the robot through each point and broadcast a success or fail completion message for each way-point. The completion message triggers the next way-point in the list to be sent to the navigation system. The control node will capture the completion message as well and take a sample at each successful way-point as discussed in Section 4.2.3.

4.2.3 Control Node Autonomous Sampling

The control node is responsible for many different tasks in the ARMR. One of the major tasks it does is the coordination of movement and measurement. When the robot is set for autonomous sampling in the GUI, the control node will intervene to
take measurements. The exploration node, which ever is in use, will continuously attempt to send new goals as soon as the previous goal has been reached. The control node will halt movement at the end of each goal to take a measurement. This is accomplished by using interrupts which listen for the completion of both move_base and exploration goals. A simplified version of the control node’s process is shown in Figure 4.7.

![Figure 4.7: Control node basic algorithm.](image)

The control node is only responsible for maintaining position while the radiation measurement is being taken. The exploration algorithm publishes goal positions to move_base to move the robot. Move_base sends a signal when movement is completed. This signal is used to trigger a sample measurement to be taken. The control node instructs the source localization subsystem to start a measurement of a specified
number of measurements set through a parameter which are then averaged. While the source localization system is measuring, the control node continuously publishes a zero velocity message on a higher priority topic than move_base. This will hold the robot stationary until the measurement is finished. Using an external node from the exploration algorithm allows the use of different algorithms to move the robot without needing to modify them to work with autonomous sampling. Finally, at the end of each sampling action, the control node receives the position of the sample from the source localization subsystem and publishes a marker to rviz for visualization.

It should be noted that it is possible that move_base fails to move the robot to the goal position. In this situation move_base sends a failure signal instead of a success. If frontier exploration is used this will end the exploration algorithm as discussed in Section 4.2.1. If map-less exploration is used the algorithm simply moves on to the next point. This action was chosen because the location of the robot when move_base fails is not predictable. The spacing values in the exploration algorithm can be adjusted to increase the number of measurements and account for the possibility of missing measurements.
4.3 Radiation Detection Subsystem

The radiation detector used in the ARMR is a $2 \times 2$ inch sodium iodide scintillation detector manufactured by Rexon. As discussed in Section 2.2 these types of detectors require a Digital Analogue Converter (DAC) in order to function. The DAC is used to both power the photomultiplier tube and to read the electric pulses from the tube. The DAC used for the ARMR is a URSA II Radiation Alert Multi Channel Analyser (MCA). This DAC produces a precise high voltage needed by the photomultiplier tube to operate correctly. The DAC simultaneously reads the signals from the detector and produces a spectrum of the energies measured.

The DAC is usually used with its own URSA II MCA software (see Figure 4.8) which controls many aspects of operating the detector. The software communicates with the DAC through a serial connection and issues commands to it. This will set the high voltage setting, the gain and threshold of the input, as well as other settings. When reading the values from the DAC the values are either requested at specific intervals or streamed continuously depending on the mode set. The software also is used to display the output of the DAC and make measurements of the spectra if necessary.

The ARMR required that the output of the detector be incorporated into the ROS framework. The source localization subsystem required only CPS but the ability to use a spectrum would be beneficial for future work. The URSA II MCA software did not have the ability to output live data or even to automatically send data after it has been recorded. The software also ran under windows whereas the rest of the system was already implemented using Linux. It was decided to create a new custom ROS node to control and read from the DAC.

In order to create the custom ROS node, the serial commands used by the URSA II
software had to be deciphered in order to control the detector. The first step was to install a virtual serial port in order to intercept the communication from the software to the DAC. This allowed the correlation of commands and their function. The timing of commands was also noted, which was useful later on to improve stability. The determination of the commands in this fashion is imprecise and difficult. Fortunately, a copy of the serial protocol was obtained from Radiation Safety Associates, Inc. with the help of Paul Steinmeyer. This accelerated development to a great degree. Each command that the DAC recognizes was implemented as a function in the new node. The functions were simplified and protected from misuse and integrated with ROS.

The final design implemented all functions as part of a library. This allows the use of the functions by other projects which do not use ROS. The ROS node implements the library to integrate it into ROS. In order to broadcast either the CPS or spectrum...
over topics, two new message types were included with the node. The node takes in as a parameters all the settings normally set by the software and issues the correct commands to the DAC to set them. Then, based on a parameter, the node will either prepare to send CPS or spectrum. The ARMR requires CPS so the appropriate topic is broadcast and the DAC is instructed to change to Geiger-Muller mode, where only counts since last inquiry are reported. The node instructs the DAC to start ramping up the high voltage to the specified value and waits for this to finish. The node can either start reading immediately (based on a parameter) or be commanded to start by another node by services implemented to start and stop readings. When the node starts taking readings in Geiger-Muller mode, a timer is started with a period of one second. At each second the node queries the DAC for the counts since last message and this value is sent over the topic as CPS. When the node ends, the DAC is first instructed to stop measuring and then to start ramping the high voltage down to zero.

This design is quite flexible and allows for other uses in the future. It also has been tested quite extensively to be completely stable. This part of the ARMR has very few responsibilities and was designed to do its job invisibly.
4.4 Source Localization Subsystem

4.4.1 Problem Statement

The ultimate task of the ARMR is to determine the locations of radiation sources as well as their strength. This can be expressed as the task of determining the sources’ parameters. The source parameters are their $x$ and $y$ positions and their radiation intensity. It is assumed that the number of sources are known and that the sources being localized will be within the area explored. If the source parameters were known, Equation (1.1) could be used to determine the flux or CPS at any point in the area from each source and the total will be the sum of the radiation from each source. To solve for the source parameters the measurements from the URSA node and the positions from the robot localization system could be used to solve the inverse problem. However, due to the dimensionality and uncertainty in the measurements, a deterministic method cannot be used.

Optimization methods are adept at solving highly dimensional problems. There are several optimization methods which are able to determine a local or global minimum of an arbitrarily high number of dimensions. The solution desired from the algorithm is the locations and strengths of each of the radiation sources. The algorithm will use a cost function in order to quantify the solutions. Since the search space for the optimization method is based on measurements with error its shape is unknown before running the source localization algorithm. Metaheuristic algorithms are a particular type of optimization algorithm that make no assumptions about the shape of the function which it is trying to optimize. This makes these algorithms well suited to the problem of source localization. The Particle Swarm Optimization (PSO) algorithm was chosen to be utilized in the source localization subsystem. The PSO algorithm, like other metaheuristic algorithms, does not guarantee a globally optimal solution.
The specific implementation of the PSO will greatly affect its performance.

4.4.2 Cost Function

In order to implement this algorithm, a radiation model and a cost function need to be defined. The algorithm will attempt to minimize the cost function in order to find a global best. The radiation model used in the cost function is based on Equation (1.1) and is given as follows: given \( i \) sources with an intensity of \( I_i \) at a distance \( R_i \) meters from the detector, the measured radiation from one source becomes:

\[
M = \sum \frac{I_i}{R_i^2}, \quad R_i = \sqrt{(X - X_i)^2 + (Y - Y_i)^2}
\]  

(4.1)

where \( M \) is the measured intensity and \( X, Y, X_i, \) and \( Y_i \) are the measurement locations and source locations, respectively. This is an application of the inverse square law for radiation modeling. This model is usually only used to transform intensities from different radii into other radii. The net effect is that it is finding the intensity at a radius of 1 m. This means that the algorithm is less accurate the closer the measurement is to the source, especially when closer than 1 m. This model also relies on other assumptions such as the radiation reaching the detector has a clear path with no shielding and that the detector is equally sensitive in all directions. These assumptions will need to be tested with actual radiation sources and so need to be tested during the full system tests.

The PSO will attempt to find the independent variables, in this case the source parameters, which minimize the cost function. The cost function should provide a metric for how well the predicted parameters fit the actual measurement data. This is accomplished by comparing the estimated measurement at each measurement location using Equation (4.1) to the actual measurement made. The cost is defined as the
RMS error between the actual and estimate measurement. The cost function is:

\[
Cost = \sqrt{\sum_{j=1}^{m} \frac{(M_{j,p} - M_{j,obs})^2}{m}}
\]  

(4.2)

where there are \( m \) observed intensities (\( obs \)) which are compared to the predicted intensities (\( p \)).

4.4.3 PSO Method

The PSO method is a population based method. The algorithm initially starts with a population of \( N \) particles uniformly distributed throughout the decision space. Each particle is a possible solution to the cost function. For the application of source localization, each particle will be a column vector with \( x, y \), and intensity values repeated for the number of sources. Each particle moves around the decision space semi-randomly. Each particle has a velocity vector which describes its motion. Its position is incremented by the velocity vector at each iteration. All of the particles work together by sharing some information about their progress. The new velocity is the weighted sum of the previous velocity, the spatial distance (in solution space) to the particle’s lowest cost solution (called \( p_{best} \)), and the spatial distance to the populations best solution (called \( g_{best} \)). The defining functions for PSO are:

\[
v_{(n+1)} = Wv_n + c_1u_1(P_{p_{best}} - P) + c_2u_2(P_{g_{best}} - P)
\]  

(4.3)

\[
P_{(n+1)} = P_n + v_{(n+1)}
\]  

(4.4)

where \( v \) is the particle’s velocity and \( P, P_{p_{best}}, \) and \( P_{g_{best}} \) are the positions of the
particle, the population’s best, and the global best, respectively. In Equation (4.3), \( W, c_1, \) and \( c_2 \) are tuning constants and \( u_1 \) and \( u_2 \) are random variables between 0 and 1. It is possible to see that the farther a particle is away from either \( pbest \) or \( gbest \) the stronger it is pulled toward them, respectively. This causes the particles to pull toward the \( gbest \) if they are initially far away in the search area from \( gbest \). The \( W \) is called the inertial value. It encourages the particles to continue on in their current direction. This adds some randomness to the algorithm and makes sure that it will explore a larger area around the personal and global best.

To run the algorithm, a random population of a specified size was generated and each particle would go through a set number of updates with cost function checks at each stage. In order to initialize the population each parameter for each source needs to be randomly generated. For each source the \( x \) and \( y \) position and its intensity must be initialized. The position values are uniformly distributed in the area spanned by the measurement observations inflated by 15%. The intensities are uniformly distributed between 0 and \( 10^7 \) in order to ensure the search space will cover all expected values.

This standard configuration is not robust to different sizes of data or different starting conditions. In order to improve the performance and increase the likelihood that a global minimum is found, a few changes were made to the standard PSO algorithm.

### 4.4.4 Testing and Modifications to Base Algorithm

Throughout each stage of development the PSO was tested with the measurement data from McDougall’s et al. [8] testing. This way the performance of each change could be measured. This allowed the algorithm to be tuned to be efficient without the need to obtain radiation measurements. Using the obtained data, the parameters
were determined: $c$ values of 1.49, and a $W$ value of 0.72.

Figure 4.9: A comparison of the MCMC results from McDougall et al. [8] and the results of different neighborhood topologies used in the PSO algorithm.

In order to improve this algorithm, the way that the population communicates was adjusted. In standard PSO, all particles are aware of every other particle. This causes the population to converge quickly, but possibly skips over a better solution. The first change implemented was to limit each particle to only be able to communicate with specific neighbors. Figure 4.9 shows the comparison of the results of different neighborhoods to the MCMC algorithm used in McDougall’s et al. work [8]. It was determined that the “mesh” or “grid” connected neighborhood performed the best and was very consistent as well. In this configuration the particles can only communicate with the particles directly adjacent in either the vertical or horizontal direction. The particles will still converge to a single point but will take longer to do so. This is ideal for discontinuous functions because a minimum may be isolated and difficult to find. It also allowed the use of fewer particles, improving the performance of the algorithm. Over 30 runs, the mesh connected PSO missed the minimum cost solution four times and the worst solution found was only 0.2 $m$ total RMS error higher than the minimum.
The second change was to add a stopping function. This allowed the algorithm to be set with more iterations than necessary without wasting time circling the global best or improving the global best beyond a necessary amount. This is accomplished by determining the distance of the best 10% of population to the global best. This is accomplished with a partial sort of the total population based on personal best. A partial sort is used to improve performance of large populations. The total RMS distance from the global best is then summed up. If the total is less than 0.02 the algorithm is stopped. This will prevent the algorithm running unnecessarily after a high degree of precision has been met.

During preliminary testing the number of particles was tuned to work well with this set of data as well. The values of 100 particles and 1,000 iterations or 1,000 particles and 100 iterations showed promise as being the most efficient combination. This proved to make the algorithm perform very well and fast for the test data that was obtained from McDougall et al. [8]. The combination of 100 particles and 1,000 iterations was chosen as the final settings. This allowed the algorithm to get a very high degree of repeatability and would only take 30s to complete. However, this proved problematic for other sets of data. When the first set of testing was performed indoors, the results were far off from the results previously obtained. The priorities of the algorithm were reevaluated and retuning the algorithm for reliability was investigated. The goal was to find ways to improve the likelihood that a global best would be found. The number of particles was increased to 250 with 3,000 iterations. The higher number of particles and iterations meant that the algorithm would take much longer to run but the stopping condition would cap the total time taken. Another method was used to further increase the reliability of the algorithm. This involves reseeding the population after it converged. This was accomplished by rerunning the algorithm with the previous global best copied to the new population. The rest of the population would be generated randomly as usual. This should increase the area
traveled by the particles. The stopping condition for when to stop reseeding is when the same global best is found 10 times in a row. Combined with the original stopping condition, this effectively increases the number of particles and iterations without the additional computational cost of actually increasing them. This guarantees that the algorithm converges to some solution and the possibility that it is a global best is greatly improved. The results of this improved algorithm are seen in Chapter 5.

4.4.5 ROS Integration

The source localization subsystem had to be usable within the ROS framework in order to communicate with other nodes in the system. So a processor node was created to control the PSO. The source localization subsystem primarily communicates with the control node in order to take new samples and to run the algorithm. Sampling actually occurs within the processor node. The control node instructs the processor node to take a sample for a specified amount of averaging time through a ROS action which the processor node provides. This triggers the processor node to start listening to the topic which the URSA node is broadcasting. The measurements are averaged and then the position is based on the transform frame attached to the counts message and then determined by the TF system. This data is added to the cost function. The processor node also provides another action which is used to run the PSO. The number of particles and iterations as well as the number of sources is determined by parameters within the action set by the control node. The PSO is run and the result is published back to the control node, which displays the result in the GUI. The details of the process are also printed to the console for analysis. A final interface exists in the processor node. It is a service which allows the current samples to be erased. This is used by the GUI to reset the samples by a button.
4.5 Displays and Controls

4.5.1 Graphical User Interface

The GUI used with the ARMR simplifies the controls of the robot. Through the GUI the user can have full control over the ARMR. The GUI also is the primary way to view the current position and progress of the robot. The outputs of each of the different subsystems are displayed in the centre of the GUI. Those controls are contained in a panel on the right hand side. Figure 4.10 shows the GUI a user sees. The centre of the screen shows the current position of the robot, the output of the LiDAR, an exploration polygon, the beginnings of the live intensity map, and an incomplete map. The right hand side is the control panel for the ARMR.

The GUI is made up of three main component types: tools, displays, and dockable panels. Each component is an addition to the standard ROS visualiser rviz. Rviz by
default is a quite effective robot viewing tool and is quite useful for testing different navigation tools. It has display types for viewing 3D models of the robot, the different transforms, including the movement transform, and the output of different sensors. It also has tools for sending navigation goals and measurement. Most importantly, \texttt{rviz} allows the creation of each type of component through standard interfaces. This allows the development of custom components which makes using the ARMR much more streamlined.

The only tool required by the ARMR is the point tool. This is a default tool built into \texttt{rviz} and publishes a point in the fixed frame of the view. The exploration subsystem uses these points to generate a polygon area in which to explore. Figure 4.10 shows a completed polygon in red in the centre of the screen. The other useful tool is the 2D Nav goal tool. This tool allows the user to publish a navigation goal for the navigation subsystem to navigate to. This is useful when moving the robot a long distance when not sampling. When moving a short distance the thumb-stick can be used on the control panel.

A dockable panel was created to implement many of the controls of the ARMR. This control panel interfaces with many of the subsystems in order to instruct them during use. Starting from the top there are two displays for live data from the ARMR: the current battery level and the CPS reported by the radiation detector. Below those is a toggle to switch between indoor and outdoor use. Below that is the navigation subsystem controls. First, either AMCL or gmapping must be selected. If AMCL is desired, first a map needs to be selected from the drop-down which is automatically populated with the available maps located on the control computer. When either is selected the navigation system is started and the robot can be moved with navigation goals. To the right of these buttons is a button to stop the navigation system and one to start the exploration system. The exploration system is also stopped when the navigation system is stopped. Below the navigation controls are two toggles which
send signals to the control node. The first instructs the control node to take samples autonomously. This causes the control node to start a new sample each time the robot arrives at a waypoint. The second toggle enables and disables the live radiation intensity map (see Section 4.5.2). Below those toggles are the controls for the source localization system. First is a drop-down to select the number of sources. Second is a button for executing the source localization. Below the drop-down is a button for manually starting a sample if the user would like to add a measurement. The next two buttons were installed to test different expansions to the source localization system which were not implemented. Finally there is a button to reset all the samples if the robot is moving to a new area or an area needs to be retested. Below the source localization controls are motion controls. On the right there is a thumb-stick to control the robot. The thumb-stick has a high priority and can be used to prevent unwanted movement from the navigation system in a pinch. This would be an easy way to control the robot if the control computer was a touch screen tablet. Beside the thumb-stick is a software emergency stop button. This button supersedes any motion command and shuts down the navigation system. At the bottom of the control panel is a button to save the map and a button to shut down the ARMR.

The display types used for the ARMR are almost all included with rviz. The navigation and exploration system uses standard displays for the maps and lines. The control node uses a marker display to mark each measurement. The robot model is published by a node developed by Clearpath Robotics and is displayed using a standard display type. The live radiation intensity map is displayed using the built in cost map display normally used with the navigation system. The only custom display type is the AerialMapDisplay display type for displaying aerial maps for large outdoor mapping scenarios. This allows the results to be displayed relative to a larger map. Figure 4.11 shows what a finished run might look like with the ArealMapDisplay turned on.
4.5.2 Live Radiation Intensity Map

The purpose of the radiation intensity map is to display what the robot is measuring in an easy to understand way which is also persistent. In order to fill in the area of the map an inverse distance weighting method is applied to expand each measurement over an area around the robot. The results of the inverse distance weighting is stored in a cost-map where each pixel is updated with the inverse weighted average of its current value to the new measurement.

First a new measurement is received by the algorithm. It is checked as to whether it is further than 30 cm. If it is not, it is ignored. The measurement is then scaled to the resolution of the cost map. In this case it is 255 levels from 0 to 254. With a current measurement of `measurement` and a maximum CPS of `maxCounts` the equation is:
\[
\text{currentCost} = \begin{cases} 
\frac{255 \times \text{measurement}}{\text{maxCounts}} & \text{for } \text{measurement} < \text{maxCounts} \\
254 & \text{for } \text{measurement} \geq \text{maxCounts}
\end{cases} \tag{4.5}
\]

where \text{currentCost} is the cost used for updating the cost-map. Before applying the new value an update area around the robot’s current location is set to ±5 meters in each direction. Each pixel’s position is determined using the spatial resolution of the cost-map which is inherited from the \text{gmapping} map or from a parameter. The weight is the inverse of the euclidean distance from the robot to the pixel raised to a power \(p\):

\[
w = \begin{cases} 
\frac{1}{d(x,x_{\text{pixel}})^p \times 3} & d(x,x_{\text{pixel}}) > \text{res} \times 4 \\
\infty & d(x,x_{\text{pixel}}) \leq \text{res} \times 4
\end{cases} \tag{4.6}
\]

here \(d\) represents the euclidean distance of the two positions \(x\) and \(x_{\text{pixel}}\) which are the positions of the robot and the current pixel respectively. The weight \(w\) is calculated as the inverse of the euclidean distance to a power then multiplied by three. The power can be set by a parameter and the number three was chosen to scale the effect of the power appropriately. The weight is ignored when the distance is less than \(4 \times \text{res}\) to avoid the explosion of the weight value. In this case, the weight is not actually set to infinity as in Equation (4.6), but ignored in the averaging equation. The averaging is then:

\[
\text{newCost} = \frac{(w \times \text{currentCost}) + (5 \times \text{pixelCost})}{w + 5} \tag{4.7}
\]

where \text{newCost} is the new cost applied to the pixel, \(w\) is the weight, \text{currentCost} is the scaled cost from the detector, and \text{pixelCost} is the current value of the pixel. This is just a weighted average of the current value with a new value weighted using inverse distance weighting. The added weight of five to the current cost of the pixel helps reduce the excessive over weighting of the new value due to the fact that the distance used in Equation (4.6) is in meters and is a relatively small value.
The benefit of using the existing pixel values against each new measurement is that
the measurement’s value and position do not need to be tracked in memory. In this
method each new measurement can be treated as an isolated event and the overall
map will still be maintained.
Chapter 5

Experimental Results and Discussion

In order to test the effectiveness of the completed ARMR, different tests must be conducted to test all the functions of the ARMR. Both indoor and outdoor tests of the navigation, exploration, and localization system were conducted. The radiation detection and source localization systems were tested for single and multiple sources to test their accuracy.

5.1 Indoor Tests

The indoor experiments were conducted in order to test the effectiveness of the navigation and exploration subsystems in an environment which has little room to explore in. The exploration algorithm described in Section 4.2.1 will be used to explore the area inside a room. The navigation subsystem will produce a map while the robot explores the unknown environment. This map will help any personnel navigate through the environment after the robot has finished, the robot having provided both a knowl-
edge of the paths available and the radiation in the area. Multiple runs of each test with different sample spacings were conducted to see the effect different number of samples had on the results from the source localization subsystem.

A test area was constructed in the Mechatronic and Robotic System Laboratory at the University of Ontario Institute of Technology (UOIT). Figure 5.1 shows the proposed setup. A command centre was set up at the west end of the lab and the test area was enclosed by a safety net.

Two tests were proposed, the first had two locations with two sources near each other and the second had one source. The first test tested the ability of the source localization subsystem to determine the location of two sources of different strength which were relatively close together. In between the sources, wall sections made from standard building materials served as obstacles and shielding for the different tests.
5.1.1 Experiments

The first configuration was set up in the test area. A platform was placed at 3 m east and 3 m south of the northwest corner of the test area. This position will be referred to as Position 1. The second platform (Position 2) was placed at 5 m east and 4 m south of the northwest corner of the room. Three wall sections were placed, dividing the area into an “S” shape (see Figure 5.2). This configuration had two 1.00 mCi Cs-173 sources placed at Position 1 and one 1.00 mCi at Position 2.

The second configuration had only one source. One platform was moved to 3 m east and 4 m south of the northwest corner of the room. The wall sections were also moved to surround the new source location. All three 1.00 mCi Cs-173 sources were placed on the platform. Figure 5.3 shows the source location and wall placement for Configuration 2. Three tests were run for each configuration. Each run was completed with a different spacing for measurements.

After a configuration was set up, the ARMR was prepared to run. The robot was
first switched on and moved into position near the entrance of the test area. The sampling distance could then be set in the exploration parameters. The navigational system was then started by selecting indoor mode in the GUI and selecting the “start gmapping” button. The navigation system then could be used to move the robot if need be. Before commanding the robot to start the trial, the sampling mode was changed to automatic mode, also through the GUI.

Using the traffic cones visible in Figure 5.2, an exploration perimeter was drawn based on the view of the laser visible in the GUI. Once the start location was selected, the robot started moving through the different positions produced by the exploration algorithm. While the robot was moving through the environment the intensity map was updated giving immediate insight into the environment being searched. From this point to the end of sampling, the ARMR operated autonomously. At the end of each test the robot returned to its start location then the PSO algorithm would be run by selecting the button in the GUI.

In Figure 5.4 the sample positions selected by the exploration algorithm for the dif-
ifferent sample spacings for each configuration is shown. The border is visible as a dark band all the way around, showing an area which should be avoided. The wall in the middle of the area is also visible as an area to avoid for exploration. This makes sure that the robot stays within the area of interest and also does not try to explore too close to obstacles. Each view is visibly rotated as well. This is due to the lack of a fixed datum indoors as each run is started without any prior knowledge of the configuration of the room. This makes the physical map produced by the navigation subsystem invaluable for understanding and verifying the output. The frontiers are visible as the border between white and grey. In each sub figure the next frontier which will be navigated to is visible as well. The green dots represent sample positions which have already been visited and recorded. The different spacings selected are not transferred exactly as the spacing obtained from the exploration algorithm. This is due to limited information at each stage. Open areas can be obfuscated early on and later discovered by placing an extra point near a previous point that is closer than the spacing value. Figure 5.4e shows a case where the algorithm is not efficient. The next frontier to be visited is across an area which has already been visited. This is expected behavior and not terribly detrimental. However, if the area was much larger with many forking paths, the efficiency may become an important factor. Finally, visible in Figure 5.4 is the current live radiation intensity map overlaid on top of the exploration costmap. Later, a detailed view of the live intensity heat map will be discussed as the user will see it without the exploration costmaps. Here, the heatmap can be seen as recorded in the middle of a test. A red area can be seen near the source in the bottom left of the “U” shaped wall section. The rest of the area is generally lower in intensity verifying that the live radiation intensity map does indeed provide useful information before exploration is even complete.

For each configuration three spacings were specified: 1.3, 1.0, and 0.75 meters. Each spacing affects how the ARMR explores the area. The time taken for each run was
noted after the ARMR completed autonomous sampling. Table 5.1 shows the complete amount of time taken for each test including sampling time and time moving in between points. The averaging time for each sample was 30 seconds. It then follows that the majority of time is spent recording measurements. If completion time was paramount, the averaging time could be reduced or the sample spacing could be increased.

Table 5.1: A comparison of the time taken to explore the area using different numbers of samples.

<table>
<thead>
<tr>
<th>Test</th>
<th>Sample Spacing (m)</th>
<th>Number of Samples</th>
<th>Exploration Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config 1, 1</td>
<td>1.3</td>
<td>15</td>
<td>9.3</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>1.0</td>
<td>24</td>
<td>14.1</td>
</tr>
<tr>
<td>Config 1, 3</td>
<td>0.75</td>
<td>46</td>
<td>25.9</td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>1.3</td>
<td>13</td>
<td>11.7</td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>1.0</td>
<td>24</td>
<td>16.8</td>
</tr>
<tr>
<td>Config 2, 3</td>
<td>0.75</td>
<td>34</td>
<td>21.6</td>
</tr>
</tbody>
</table>

5.1.2 PSO Source Localization Results and Discussion

After each measurement was taken and the ARMR moved away from the area of interest, the PSO could be run. On the test day the PSO algorithm was incomplete. The algorithm had only been tested with one data set and the results obtained during the test were not acceptable. Based on the observations on the day of the test the changes to the algorithm described in Section 4.4 were implemented. The measurements were recorded, and then processed using the complete algorithm at a later time. These results are discussed here.

First, the amount of time that the algorithm took was noted. The algorithm was run on a second generation Intel i7 mobile processor. Configuration 1 with two sources was clearly more complex, taking significantly more time than Configuration 2 to complete. Additionally, the test with fewer samples caused the algorithm to take much longer
than the test with the most samples. The total time for configuration one test one was about seven minutes. This test caused the reseeding algorithm to restart its count to ten several times, causing the extended run time. For Configuration 2, each test took a similar amount of time, averaging around 45 seconds. This is due to the fact that each run found the global minimum in the first count and took ten runs exactly. This leads to the conclusion that even the largest spacing included enough measurements for the algorithm to easily find the source parameters. However, increasing the number of sources by just one, significantly increased the difficulty of the problem for the PSO algorithm.

Table 5.2: A comparison of the time taken against different number of sources and samples for the source localization algorithm.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of Samples</th>
<th>Execution Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config 1, 1</td>
<td>15</td>
<td>424.28</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>24</td>
<td>100.42</td>
</tr>
<tr>
<td>Config 1, 3</td>
<td>46</td>
<td>70.84</td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>13</td>
<td>48.05</td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>24</td>
<td>41.97</td>
</tr>
<tr>
<td>Config 2, 3</td>
<td>34</td>
<td>48.59</td>
</tr>
</tbody>
</table>

The results of each run overlaid onto the maps made during the test can be seen in Figures 5.5 to 5.10. On each image are markers for the actual positions of the sources (red stars) and the results of the runs (green pluses). The reference locations were located based on the position of the platforms in the produced maps. The positions from the PSO are hard to differentiate from the reference locations on many of the figures due to their proximity to each other. This indicates that the accuracy is quite high. The exact results are shown in Table 5.3.

Table 5.3 shows the detailed source positions produced by the PSO algorithm. The error displayed is the RMS error for each position. With more measurements the error tends to decrease. The first test for Configuration 1 shows the highest error.
Table 5.3: A table displaying the RMS error of each test when compared to the reference location.

<table>
<thead>
<tr>
<th>Test</th>
<th>Position 1</th>
<th>RMS Error</th>
<th>Position 2</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$ (m)</td>
<td>$y$ (m)</td>
<td>$x$ (m)</td>
<td>$y$ (m)</td>
</tr>
<tr>
<td>Reference</td>
<td>4.16</td>
<td>0.04</td>
<td>4.26</td>
<td>-2.08</td>
</tr>
<tr>
<td>Config 1, 1</td>
<td>4.1243</td>
<td>-0.1282</td>
<td>0.1719</td>
<td>4.2781</td>
</tr>
<tr>
<td>Reference</td>
<td>4.22</td>
<td>-1.02</td>
<td>3.66</td>
<td>-3.04</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>4.2656</td>
<td>1.0573</td>
<td>0.0589</td>
<td>3.5719</td>
</tr>
<tr>
<td>Reference</td>
<td>-0.62</td>
<td>-3.60</td>
<td>-0.14</td>
<td>-1.58</td>
</tr>
<tr>
<td>Config 1, 3</td>
<td>-0.5082</td>
<td>-3.6761</td>
<td>0.1352</td>
<td>-0.1729</td>
</tr>
<tr>
<td>Reference</td>
<td>2.52</td>
<td>-1.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>2.5708</td>
<td>-1.3202</td>
<td>0.1491</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>2.66</td>
<td>-2.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>2.7323</td>
<td>-2.3455</td>
<td>0.0725</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>3.22</td>
<td>-2.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 2, 3</td>
<td>3.2374</td>
<td>-2.4679</td>
<td>0.0510</td>
<td></td>
</tr>
</tbody>
</table>

with 0.17 m for one of the source positions. This amount of error is still less than the diameter of the platforms used. The reference locations were found by analyzing the map produced by the navigation subsystem. The map has a resolution of 0.02 m per pixel and imperfections in the map may contribute to an offset of a few pixels in the reference locations. The error for the tests with the most measurements for Configurations 1 and 2 both had errors which were on the order of a few pixels in terms of the map and only a few centimeters overall.

Table 5.3 represents the predicted position of the sources. The algorithm also reports the assumed CPS at one meter from each source. Table 5.4 shows the output from the PSO algorithm. It is clear that for Configuration 1 the radiation intensity at Position 1 is about twice that of Position 2 which is what was expected. Configuration 2 shows an intensity roughly the sum of the CPS of the first tests. This result shows that the PSO algorithm is producing reasonable and accurate results for each source.
Table 5.4: Source intensity predictions for the indoor trials.

<table>
<thead>
<tr>
<th>Test</th>
<th>Position 1 (CPS at 1 m)</th>
<th>Position 2 (CPS at 1 m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config 1, 1</td>
<td>11,246</td>
<td>6,052</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>11,325</td>
<td>5,384</td>
</tr>
<tr>
<td>Config 1, 3</td>
<td>12,397</td>
<td>4,697</td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>16,976</td>
<td></td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>17,090</td>
<td></td>
</tr>
<tr>
<td>Config 2, 3</td>
<td>16,017</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.4: Indoor exploration operation for different sample spacings.
Figure 5.5: Estimated source locations for Configuration 1 with 1.3 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red stars).
Figure 5.6: Estimated source locations for Configuration 1 with 1.0 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red stars)
Figure 5.7: Estimated source locations for Configuration 1 with 0.75 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red stars).
Figure 5.8: Estimated source location for Configuration 2 with 1.3 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red star).
Figure 5.9: Estimated source location for Configuration 2 with 1.0 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red star)
Figure 5.10: Estimated source location for Configuration 2 with 0.75 meter spacing, produced by the PSO algorithm (green pluses) with the reference inserted (red star).
5.2 Outdoor Tests

The outdoor experiments were conducted in order to test the effectiveness of the localization system in an unknown outdoor environment, without useful obstacles to localize from. The baseball diamond located at the University of Ontario Institute of Technology (UOIT) was chosen for this purpose. The area is suitably flat for the size of the ARMR and also provides a challenge for the navigation system due to the ease of slipping on the soil surface.

A location for the base of operations was chosen and a base reference location for the RTK GPS system was chosen. This was decided to be the north east end of the diamond with the GPS location being centered on one corner of home plate. The command centre and reference GPS can be seen in Figure 5.11. The reference location was surveyed using the base station GPS receiver and a smart-phone. A logging time of 45 minutes was used to allow a high degree of accuracy in the post processing. The GPS logs were downloaded from the GPS unit and uploaded for analysis. The output from the Precise Point Positioning (PPP) analysis can be seen in Appendix A.

A 21 m square area was measured within the infield diamond southwest of the base location. Pylons were placed at the corners and were intended to be used as reference points for defining the area to be searched. These proved difficult to view in rviz and the 1 meter grid in rviz was used instead to draw a 20 m grid. Two source positions were selected using the home plate-first base direction as west and home plate-third base direction as south. Using these directions the source positions were measured and marked at 10 meters south by 10 meters west of the northeast corner. From now on this position will be refereed to as Position 1. The second position (Position 2) was marked at 3 meters south by 5 meters west of the northeast corner of the area. A view of the two source locations can be seen in Figure 5.12. In order to maintain accuracy, the source positions were then surveyed in the same manor as the base
location with a logging time of 20 minutes. For the full output of the PPP analysis see Appendix A.

The UTM measurements in north-east format and the associated sigmas of each surveyed position can be seen in Table 5.5.

Table 5.5: The UTM positions of the surveyed locations of the source locations and base location.

<table>
<thead>
<tr>
<th>Position</th>
<th>UTM (North) Zone 17</th>
<th>Sigmas (95%)</th>
<th>East (m)</th>
<th>Sigmas (95%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Location</td>
<td>4867438.263</td>
<td>0.127 m</td>
<td>668999.961</td>
<td>0.102 m</td>
</tr>
<tr>
<td>Position 1</td>
<td>4867425.647</td>
<td>0.167 m</td>
<td>668993.234</td>
<td>0.152 m</td>
</tr>
<tr>
<td>Position 2</td>
<td>4867434.230</td>
<td>0.162 m</td>
<td>668995.661</td>
<td>0.148 m</td>
</tr>
</tbody>
</table>

5.2.1 Experiments

Four tests were conducted in the outdoor test environment. Two different source configurations were tested. The first configuration had two of the 1.00 mCi Cs-137 sources placed at Position 2 and one source at Position 1. The second configuration
had all three of the sources placed at Position 1. The first configuration is designed to test the localization algorithm’s ability to localize multiple sources with different levels of radioactivity. The second configuration is designed to test the live intensity map at further distances as well as the localization system with one source. For each source configuration, two different row spacings were tested to determine the effect of more samples in the same area. The goal aliasing was kept at 3 m but the row spacing was changed from 3 m to 2 m between tests. Changing the row spacing or goal aliasing is accomplished by changing the parameter in a file on the base station computer.

Before beginning each test the compass was calibrated by running a calibration run from the base station computer. This rotated the robot slowly in place while recording magnetic readings. This aligns the compass with the local magnetic field and reduces compass drift during trials. The navigational system was then started by selecting outdoor mode in the GUI and selecting the “start gmapping” button. The navigation system then could be used to move the robot if needed. Before commanding the robot
to start the trial, the sampling mode was changed to automatic mode, also through the GUI.

To start a trial, the ARMR was commanded to search a 20 m square area by drawing the bounding box in rviz on the base station computer and selecting the start location. Once the start location was selected, the robot would immediately start moving through the different positions planned by the exploration algorithm. While the robot was moving through the environment, the intensity map was updated giving immediate insight into the environment being searched. From this point to the end of sampling the ARMR operated autonomously. At the end of each test the robot returned to its start location. Then the PSO algorithm was run by selecting the button in the GUI.

The sample positions selected by the exploration algorithm for each test are shown in Figure 5.13. In this figure the axes show the distance in meters from the base station which was located in the north east corner of the diamond. Therefore the points lay south and west of the base station. The coordinates are placed in this fashion due to the ROS standard of having the x-positive direction face north and the y-positive direction face west.

As is visible in Figure 5.13 the positions are not rigidly aligned to a grid. As well the actual positions were recorded instead of commanded positions to account for tolerance in the navigation system and settling time of the GPS sensor as discussed in Section 4.1.

The navigation system created a gap in the measurement locations as can be seen in Figure 5.13b and 5.13d. This is due to the fact that the requested location was too close in proximity to Position 1. The navigation system determines that the location is unreachable and the position is skipped. For more information on this behavior see Section 4.2. Our results suggest that more dense measurement parameters cause
more failed measurements to occur.

While the area that the ARMR searched was a fixed size, the number of points that were reached changed based on how the bounding box was drawn as described in Section 4.2, and also which points were reachable. This resulted in each test having a different number of samples. The time it took for the robot to finish was worth noting as well. Table 5.6 shows the number of samples each test recorded compared to the total run time. The results show a fairly consistent \( \sim 28 \) seconds per sample. Considering that each position had an averaging time of 20 seconds it then follows that the travel time was under 10 seconds per sample.

Table 5.6: The time taken to search a 20 m square area.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of Samples</th>
<th>Time (Excluding Travel Home) (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config 1, 1</td>
<td>51</td>
<td>23:10</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>87</td>
<td>40:10</td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>64</td>
<td>30:04</td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>81</td>
<td>36:33</td>
</tr>
</tbody>
</table>

5.2.2 PSO Source Localization Results and Discussion

Once the ARMR completed sampling, the PSO can be run. The algorithm was run by first specifying the number of sources using the drop down in the GUI. Then the PSO was run using the appropriate button in the GUI. This will run the algorithm described in Section 4.4. The algorithm was run in the background with the main program still running. The computer has a second generation Intel i7 mobile processor.

First the algorithm’s run time was noted. The run time difference for different numbers of samples was negligible, a maximum of around 10 seconds. Increasing the number of sources, however, significantly increased the processing time. The time changed from an average of 1:02 minutes to an average of 3:26 minutes. The added
Figure 5.13: Sample locations determined by exploration algorithm for Configuration 1 (a,b) and Configuration 2 (c,d).

complexity of more sources clearly takes longer to compute.

Initially the PSO results look accurate when viewing the results on the same scale as the search area as in Figure 5.14. After the experiments were complete, the results were analysed further.

The positions measured with the GPS and described in Table 5.5 were used to verify the results of the tests. In order to relate the UTM positions to the results from the PSO algorithm the UTM measurements had to be zero referenced to the base location. The base location was also used as the world frame axis in the localization system and for the source localization algorithm. However, the base location also had
Figure 5.14: Estimated source locations produced by the PSO algorithm (crosses or squares) with the GPS reference inserted (red stars).

to be offset due to the issues discussed in Section 4.1.2.5.

Using the reference location and the results produced by the PSO algorithm, the positions could be plotted as seen in Figure 5.15. The magnified view shows that the estimates are in fact positioned near the reference locations but with some offset. The exact error is discussed below.

Table 5.7 lists the detailed source positions produced by the PSO algorithm. The error displayed is the RMS error for each position. It is clear that with more sources the total error increases. Less desirably, the individual errors also tend to increase with multiple sources. The error for Configuration 2 with one source was less than 1
Figure 5.15: Sample locations determined by exploration algorithm.

meter RMS, a very reasonable result. The Configuration 1 error is slightly higher with a per source RMS error around 1 meter. The tests with more samples seem to result in a higher total error as seen in Table 5.7. The error in the reference measurement and instantaneous GPS error during the test should be considered when viewing these results. For instance, the RMS error of the base station position was 0.163 m and the error for Position 1 was 0.226 m. These values represent highly accurate measurements which were the result of post processed data from long stationary recordings. Assuming these positions are correct, the results seem to be skewed in one direction, appearing to possibly be rotated around the origin, which is off of the bottom right corner of Figure 5.15.

Figure 5.16 shows the live radiation intensity map, the PSO results, and the GPS
Table 5.7: The RMS error of each test when compared to the GPS reference.

<table>
<thead>
<tr>
<th>Test</th>
<th>Position 1</th>
<th>RMS Error</th>
<th>Position 2</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x \ (m)$</td>
<td>$y \ (m)$</td>
<td>$x \ (m)$</td>
<td>$y \ (m)$</td>
</tr>
<tr>
<td>Reference</td>
<td>-16.4535</td>
<td>6.9396</td>
<td>-7.8704</td>
<td>4.5126</td>
</tr>
<tr>
<td>Config 1, 1</td>
<td>-16.8423</td>
<td>5.8850</td>
<td>1.1240</td>
<td>-8.2908</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>-17.1756</td>
<td>6.2933</td>
<td>0.9691</td>
<td>-8.5704</td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>-16.2191</td>
<td>6.7112</td>
<td>0.3272</td>
<td></td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>-16.6615</td>
<td>6.3620</td>
<td>0.6139</td>
<td></td>
</tr>
</tbody>
</table>

reference on one image. The red spots created by the intensity map, indicate the path the ARMR took while navigating, clearly driving around the platform that the source was placed on. The ARMR then took a sharp turn in order to make the next navigation waypoint. This indicates that the actual source position, as measured by the ARMR, does closely match the PSO result. This indicates that the issue is with the localization system. The offset errors in `navsat.transform_node` discussed in Section 4.1.2.5 prompted an investigation into the node’s possible contribution to a further offset. It was found that the magnetic orientation was used extensively to orient the robot to the UTM grid. Most significantly, a measurement is taken at the start of the node and used to rotate all positions produced while the node is running. It is possible that a temporary offset just after starting could translate into significant error the farther the ARMR travels from the datum. Further testing would need be conducted with a more accurate magnetic sensor or using a different GPS translation node or both.

Table 5.7 only shows part of the results. The algorithm also reports the assumed CPS at one meter from each source. Table 5.8 shows the output from the PSO algorithm. It is clear that for Configuration 1 the radiation intensity at Position 2 is about twice that of Position 1 which is what was expected. Configuration 2 shows an intensity roughly the sum of the CPS of the first tests. This result shows that the PSO algorithm is producing reasonable results for each source.
Despite the error in localization the overall accuracy of the results is excellent. The source positions provide and intensities provide sufficient accuracy to guide a first responder directly to the source. These experimental results prove the ARMR is an effective radiation mapping tool.

Table 5.8: The source intensities reported by the source localization algorithm for the outdoor tests.

<table>
<thead>
<tr>
<th>Test</th>
<th>Position 1 (CPS at 1 m)</th>
<th>Position 2 (CPS at 1 m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config 1, 1</td>
<td>6345</td>
<td>10751</td>
</tr>
<tr>
<td>Config 1, 2</td>
<td>5733</td>
<td>11961</td>
</tr>
<tr>
<td>Config 2, 1</td>
<td>14076</td>
<td></td>
</tr>
<tr>
<td>Config 2, 2</td>
<td>16382</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Live View and Intensity Map

While the test is running the intensity map is updated continuously. The maximum intensity is represented with red and the lowest value is represented with blue. The maximum value can be set through a parameter as discussed in Section 4.5.2. Figure 5.17 shows an example of the graphical view in the middle of a test. In the figure it can be seen where the ARMR has reported a high CPS near the centre of the search area. On the parallel paths it is also visible that there is a slight shift toward red. In areas that are at or near background CPS the ARMR draws a narrow blue trail. This is due to the distance weighting operation of the intensity map and the fact that a zero value represents transparent in the display. In areas of low CPS, the combination of distance weighting against the existing zero value and integer rounding still results in a zero value surrounding the ARMR. This has the added benefit of making areas which have valuable information more visible, such as the area near Position 1 in the centre of the search area.
Figure 5.17: An example of the GUI view during a test
5.4 Discussion

The ARMR completed each test successfully. The accuracy and precision has proven to meet and exceed requirements. This section discusses the ARMRs operation, practical considerations, and possible improvements.

5.4.1 Operation Procedure and Safe Handling

In order to conduct a search with the ARMR there was some preparation. Most actions necessary were conducted through the GUI. Some parameters are configurable in a parameter file. The operating procedure is as follows:

1. The ARMR first should be set down in a safe area.

2. A base of operations should be setup with the control computer and RTK GPS base station if necessary. The area of interest should be accessible by the ARMR from this area.

3. Before beginning the main application the measurement distance and averaging time should be set in the parameters file.

4. The main launch file can then be run. This will launch all necessary software nodes to operate the ARMR.

5. The GUI will be visible at this point along with the dockable panel on the right.

6. At this point the decision between indoor and outdoor operation needs to be selected with the toggle at the top of the dockable panel.

7. At this point there are two options:

   • If a map is available it can be loaded by first selecting the correct map in the dropdown menu. Then AMCL can be started by selecting the appropriate
button.

- If a map is not available then the "gmapping" button should be selected.

8. At this point the robot can be driven either by the thumbstick or by issuing a navigation goal with the appropriate tool. The ARMR should be moved close to the area of interest.

9. Once the robot is in position adjacent to the area of interest, the exploration system can be started by selecting the appropriate buttons in the GUI. The autonomous sampling toggle should also be set to on to enable automatic measurements to take place.

10. Now, using the point tool an exploration boundary can be drawn in the main view of the GUI. When the boarder changes from blue to red the border is complete.

11. At this point the autonomous sampling toggle should be turned to the on position. In this setting the ARMR will take a radiation measurement at each successful waypoint movement.

12. The last point can now be placed. This will trigger the start of the autonomous exploration and serve as a starting position and a return to point if the robot strays out of the explore boundary. When the robot finishes its exploration it will return to its start position.

13. At this point all measurements have been recorded and the PSO can be run. First, select the number of sources expected in the area from the dropdown menu.

14. Next, select the "Run Source Localization" button. The PSO will run and the results will be displayed in rviz.
15. Now that the algorithm has run the map can be saved with the save all button. Before retrieving the ARMR it is important to check for contamination. The first check should be to see if the current radiation measurement is higher than background radiation levels. This can be done by guiding the robot to an area with known background away from personnel and checking the radiation level there. The live view can also be used to see if an area previously coloured blue is recoloured after another pass with a hotter value. Appropriate decontamination should be applied to the ARMR before handling the robot.

5.4.2 Improvements

The ARMR performed well in the tests conducted, however, some improvements could be made to allow it to perform in a wider array of scenarios.

- The tests performed used radiation sources which are not intense enough to affect the internal components of the ARMR. If the system would be used in a highly radioactive environment the internal components would need to be shielded. In most use cases this would not be necessary. However, a disaster such as a reactor failure would require shielding.

- The outdoor tests were conducted outdoors in the cold. This poses a problem for the radiation detector. Scintillation detectors are known to shift the spectra with temperature changes. The detector was used in Geiger Muller mode and so the entire spectra was used for measurements, meaning that any shift in spectra did not affect the results presented here. However, if the spectra would be used in future, steps would need to be taken to account for temperature changes.

- An accurate number of sources needed to be selected in order for the PSO to produce an accurate result. In these tests the number of sources was known by
the operator but in a real-world test it would not be known. The number of sources could be guessed by looking at the live view if the sources are clearly separated. If the number of sources is under estimated the PSO will produce a solution which best fits the data. This will correspond to two of the sources being represented by one source in the solution. If the number of sources is over estimated, either there will be sources in the solution which have a very low intensity or two sources will be very close and correspond to a single real source. By rerunning the algorithm with a different number of sources close to the initial guess, it is possible to refine the initial guess. A method for automatically determining the number of sources could be an area of future work.
Chapter 6

Conclusions and Recommendations for Future Work

6.1 Conclusions

The completion of this work presented a fully autonomous radiation mapping robot. The ARMR has been successfully implemented and thoroughly tested.

A prototype system was developed using off-the-shelf hardware and the development of purposed software subsystems. Navigation, exploration, and source localization subsystems were developed, integrated, and customized to both indoor and outdoor environments. The differing requirements for indoor and outdoor operation were identified and the corresponding systems adapted.

Ease of use and information available to the operator remained a priority throughout development. The ARMR was designed to be controlled through an intuitive GUI using a visually recognizable representation of the surroundings that is true to scale. As well, easy to use drawing tools, buttons, and a thumb stick were implemented. All of the information pertinent to personnel is readily available within the GUI as
markers or as a radiation intensity map.

Requirements for further development and modularity were also considered during development. The modularity of each subsystem was maintained through the use of ROS and standard messages for integration. The source localization subsystem was also made to be modular. The PSO algorithm was developed to be independent of the cost function on which it operates. Further development of a more complex radiation model could be implemented as a new cost function.

Test scenarios were planned and conducted to evaluate both the indoor and outdoor capabilities of the ARMR. Using the user interface as an actual user would, the usability of the interface was evaluated.

The indoor trials consisted of two configurations of obstacles using realistic materials with one and two radiation source locations. Three tests were conducted for each configuration with different sample spacings. The ARMR was guided by the frontier exploration subsystem to autonomously take measurements in an area defined by the user. The navigation subsystem produced a map of the area and provided localization using LiDAR. The source localization results showed a general increase in accuracy with a smaller measurement separation with an error in the range of 10 cm per source. The measured source intensities produced by the source localization system corresponded to the relative intensities of the sources used.

The outdoor trials focused on the challenges of localization and exploration in a large outdoor environment. The outdoor trials also consisted of two configurations with one and two radiation source locations, respectively. Each test was conducted on a 20 meter square area. An RTK GPS sensor was used to provide localization for the navigation subsystem. The source positions produced by the source localization subsystem were within 1.5 m of reference locations surveyed using GPS equipment.

The ARMR has proven to be a capable tool for radiation mapping. The ARMR in its
current form could be used by first responders to improve safety by informing them of any potential hazards without exposing any humans to danger.

6.2 Recommendations for Future Work

The ARMR has been shown to be a capable radiation mapping prototype. However, the ARMR has a few areas which could be improved via future work.

- **Implement on other platforms:** The hardware required could be miniaturized allowing for the use of a wide range of robotic platforms. An autonomous amphibious robot could be used to detect radiation in more rugged terrain. The system could even be adapted for use on a Unmanned Aerial Vehicle (UAV) allowing the system to cover a large area quickly. The use of ROS would make the modifications necessary an easier task.

- **Refine outdoor localization:** The outdoor localization system preformed well during the outdoor trials. However, some offset in the results was noted. Further research into the source of the offset could lead to improved accuracy for outdoor operations.

- **More sources:** Tests with more than two sources could help verify the accuracy of the source localization subsystem. The run time of the localization system with increasing number of sources could also be analysed.

- **Develop new autonomous exploration techniques:** In order to use the ARMR in more varied situations, new control methods could be developed. One such method could be made to allow the ARMR to continuously monitor an area and report when a anomalous reading is detected, at which time it could execute the source localization system.
• **Improve detection**: A detector collimator can eliminate the detection of unwanted radiation. Research into its benefit for the ARMR could be an area of future research.

• **Incorporate additional radiation data**: The radiation detection subsystem is capable of measuring spectral information. This information could be used to differentiate between sources with different compositions. This would improve the accuracy of the results in situations with multiple different radiation sources.

• **Refine radiation model**: The radiation model used in the ARMR does not account for the shielding effect of obstacles, possibly degrading the accuracy of the system. The location of obstacles relative to each measurement is known through the map produced by the navigation subsystem. This map could be used in the radiation model to improve the results of the source localization subsystem.

• **Refine source localization subsystem**:
  
  – A source localization system which is able to automatically determine the background would improve the accuracy of the PSO or any other localization algorithm.

  – A system to automatically determine the number of sources by using the existing data would improve the ease of use of the system.

  – A source localization system which leverages known elements of the radiation model can improve performance and accuracy. Implementing an algorithm such as the one demonstrated by Chin et. al [20] could improve the run time of the source localization subsystem.

• **Refine exploration subsystem**: New exploration methods could improve the speed at which an area could be explored. A method which incorporates
the information gain for each measurement into the exploration method could improve efficiency in exploring an area.
References


[56] T. Takasu and A. Yasuda, “Development of the low-cost RTK-GPS receiver with
Appendix A

Results of PPP analysis provided by Natural Resources Canada.

A.1

Position of base station, estimated from 45 minutes of stationary GPS data.
CSRS-PPP (V 1.05 34613 )

Data Start: 2016-11-28 21:14:23.000
Data End: 2016-11-28 22:00:47.200
Duration of Observations: 0h 46m 24.20s

Apri / Aposteriori Code Std
2.0m / 2.145m

Observations
Code
L1

Frequency
Mode
Static

Elevation Cut-Off
10.000 degrees

Rejected Epochs
0.59 %

Observation & Estimation Steps
0.20 sec / 0.20 sec

Antenna Model
APC to ARP

Ant. not in PPP (0 m)
0.000 m

(APC = antenna phase center; ARP = antenna reference point)

Estimated Position for rov_201611282114.obs

Latitude (+n) Longitude (+e) Ell. Height
NAD83(CSRS) (2016) 43º 56’ 26.3959” -78º 53’ 39.1339” 110.104 m
Sigmas(95%) 0.127 m 0.102 m 0.222 m
Apriori 43º 56’ 26.557” -78º 53’ 39.246” 117.849 m
Estimated - Apriori -4.970 m 2.499 m -7.745 m

Orthometric Height
CGVD28 (HTv2.0)
95% Error Ellipse (dm)
semi-major: 1.669dm
semi-minor: 1.164dm
semi-major azimuth: -25º 58’ 24.04”

146.248 m
(click for height reference information)

UTM (North) Zone 17
4867438.263m (N) 668999.961m (E)

Scale Factors
0.99995129 (point)
0.99993401 (combined)

(Coordinates from RINEX file used as apriori position)
Modelled Tropospheric Zenith Delay (2016-11-28 21:14:23.000 GPS)
(Global Mapping Function (GMF))

Station Clock Offset (2016-11-28 21:14:23.000 GPS)

Natural Resources Canada does not assume any liability deemed to have been caused directly
or indirectly by any content of its PPP-On-Line positioning service.

If you have any questions, please feel free to contact:
EMail: nrcan.geodeticinformationservices.rncan@canada.ca
Phone: 343-292-6617
A.2

Position of the source location 10 meters south and 10 meters west of the base station, estimated from 20 minutes of stationary GPS data.
rover_201611292207.obs

**Data Start**
2016-11-29 22:07:06.000

**Data End**
2016-11-29 22:27:32.800

**Duration of Observations**
0h 20m 26.80s

**Apri / Aposteriori Code Std**
2.0m / 2.189m

**Observations**
- **Code**
- **Frequency**
  - L1
- **Mode**
  - Static
- **Elevation Cut-Off**
  - 10.000 degrees
- **Rejected Epochs**
  - 0.03%
- **Observation & Estimation Steps**
  - 0.20 sec / 0.20 sec
- **Antenna Model**
  - APC to ARP
  - Ant. not in PPP (0 m)

(APC = antenna phase center; ARP = antenna reference point)

**Estimated Position for rover_201611292207.obs**

<table>
<thead>
<tr>
<th>NAD83(CSRS) (2016)</th>
<th>Latitude (+n)</th>
<th>Longitude (+e)</th>
<th>Ell. Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>43° 56’ 25.9928’’</td>
<td>-78° 53’ 39.4499’’</td>
<td>109.804 m</td>
<td></td>
</tr>
<tr>
<td>Sigmas(95%)</td>
<td>0.167 m</td>
<td>0.152 m</td>
<td>0.359 m</td>
</tr>
<tr>
<td>Apriori</td>
<td>43° 56’ 26.007’’</td>
<td>-78° 53’ 39.426’’</td>
<td>120.082 m</td>
</tr>
<tr>
<td>Estimated - Apriori</td>
<td>-0.429 m</td>
<td>-0.532 m</td>
<td>-10.278 m</td>
</tr>
</tbody>
</table>

**Orthometric Height**
- CGVD28 (HTv2.0)
- 145.948 m
- (click for height reference information)

- **95% Error Ellipse (dm)**
  - semi-major: 2.202dm
  - semi-minor: 1.766dm
  - semi-major azimuth: -32° 7’ 3.68’’

**UTM (North) Zone 17**
- 4867425.647m (N) 668993.234m (E)
- Scale Factors
  - 0.99995126 (point)
  - 0.99993403 (combined)

(Coordinates from RINEX file used as apriori position)
Natural Resources Canada does not assume any liability deemed to have been caused directly or indirectly by any content of its PPP-On-Line positioning service.

If you have any questions, please feel free to contact:
EMail: nrcan.geodeticinformationservices.rncan@canada.ca
Phone: 343-292-6617
A.3

Position of the source location 3 meters south and 5 meters west of the base station, estimated from 20 minutes of stationary GPS data.
CSRS-PPP (V 1.05 34613)

rov_201611292227.obs

Data Start Data End Duration of Observations
2016-11-29 22:28:00.000 2016-11-29 22:49:00.400 0h 21m 0.40s

Apri / Aposteriori Code Std
2.0m / 1.662m

Observations Frequency Mode
Code L1 Static

Elevation Cut-Off Rejected Epochs Observation & Estimation Steps
10.000 degrees 0.02 % 0.20 sec / 0.20 sec

Antenna Model APC to ARP ARP to Marker
Ant. not in PPP (0 m) 0.000 m

(APC = antenna phase center; ARP = antenna reference point)

Estimated Position for rov_201611292227.obs

<table>
<thead>
<tr>
<th>Latitude (+n)</th>
<th>Longitude (+e)</th>
<th>Ell. Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>43° 56’ 26.2688’’</td>
<td>-78° 53’ 39.3313’’</td>
<td>110.385 m</td>
</tr>
<tr>
<td>0.162 m</td>
<td>0.148 m</td>
<td>0.344 m</td>
</tr>
</tbody>
</table>

Sigmas(95%)

Apriori
43° 56’ 26.323’’ -78° 53’ 39.306’’ 122.540 m

Estimated - Apriori
-1.676 m -0.565 m -12.155 m

Orthometric Height
CGVD28 (HTv2.0)

146.529 m
(click for height reference information)

95% Error Ellipse (dm)

semi-major: 2.067dm
semi-minor: 1.795dm
semi-major azimuth: -24° 50’ 16.29’’

UTM (North) Zone 17

4867434.230m (N) 668995.661m (E)

Scale Factors
0.99995127 (point)
0.99993395 (combined)

(Coordinates from RINEX file used as apriori position)
Estimated Parameters & Observations Statistics

Pseudo-Range Residuals Sky Distribution

[Diagram showing pseudo-range residuals for various PRNs]
Natural Resources Canada does not assume any liability deemed to have been caused directly or indirectly by any content of its PPP-On-Line positioning service.

If you have any questions, please feel free to contact:
EMail: nrcan.geodeticinformationservices.rncan@canada.ca
Phone: 343-292-6617