Distributed RFID Tags for Autonomous Mapping of Internal Environments Using the Robot Operating System

by

Karl Frederick Fezer

(Under the direction of Pete Bettinger)

Abstract

This thesis discusses the development of A.T.I.C.U.S.S., or Autonomous Turtlebot for Indoor Cartography, Utilizing the SCOUT System. The SCOUT System is a novel basis for indoor localization, consisting of RFID tags at the junction of carpet tiles, creating an invisible grid on the floor. Autonomous Mapping methods are developed for the Robot Operating System (ROS) in order to test the effectiveness of an RFID-based navigation system. While it is shown that RFID tags do act as landmarks, the boon to localization is slight, if at all.

Index words: RFID, SLAM, ROS, Behavior Based Robotics, Genetic Algorithms

Distributed RFID Tags for Autonomous Mapping of Internal Environments Using the Robot Operating System

by

Karl Frederick Fezer

B.A., The University of Georgia, 2009

B.S., The University of Georgia, 2010

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2014

Distributed RFID Tags for Autonomous Mapping of Internal Environments Using the Robot Operating System

by

Karl Frederick Fezer

Approved:

Major Professor: Pete Bettinger

Committee: Charles Cross
Prashant Doshi

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2014

Acknowledgments

I would certainly like to acknowledge my Thesis Committee, all of whom agreed to help me on short notice and at a great time of need. I would also like to acknowledge everyone at Interface who gave me the opportunity to test my ideas, as well as the means. I would like to thank Dr. Don Potter and Dr. Khaled Rasheed, who have given me great advice over the years, even before I was accepted into the Artificial Intelligence Program. I would also like to thank my Mother and Grandfather, who taught me from an early age that abstract thought, logic, and math could be fun.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis focuses on the development of the Autonomous Turtlebot for Indoor Cartography, Utilizing the SCOUT System, or A.T.I.C.U.S.S. This mobile platform requires Genetic Programming to learn how to use the information given by its environment as accurately as possible in order to localize using RFID tags. The following section introduces the topics which together allow A.T.I.C.U.S.S. to map an indoor environment autonomously.

1.1 Motivation

The Global Positioning System (GPS) has become the standard in outdoor navigation and localization. However, GPS does not work in many buildings due to either the number of floors or the materials the building is constructed from interfering with the radio signals from orbit. Many buildings would benefit from a navigation system as easy to use and accurate as GPS. An indoor navigation system could provide security by tracking employees and guests, by assisting visitors by providing information on where key locations or people are within the building, or by providing assistance during the time of an emergency, helping people to get out of the building quickly and emergency teams to find those still trapped inside. These applications require the integration between a map and localizer device, most likely a smart phone. Due to the proliferation of these devices, the multitude of sensors they contain, and the ease of writing custom software, they are an obvious choice. Once the environment is mapped, the map can be loaded onto the smart phone, which can act as a guide and as a tracking beacon for the person carrying it.

However, localization of personnel requires a dedicated approach to indoor mapping and navigation. Most of these direct approaches, however, run on expensive platforms running software whose sole purpose is to accurately map the inside of buildings [3, 8, 20, 23, 24, 52, 54, 56, 58, 59, 71]. These systems are complicated, cost prohibitive, and not designed for wide-spread use. One of the goals of this thesis is to map a building using open-source hardware and software on a multi-purpose platform. By using open-source hardware and software, cost is kept down and developers may easily add additional features to the robot simply, allowing more versatile applications and greater proliferation.

Interface, a global leader in carpet manufacturing, already has an RFID based robot that has been developed by SRI in order to accurately map indoor environments. However, this robot is incredibly expensive and requires many man-hours in order to map a location. The additional motivation for this thesis is to show that this can be done cheaply and autonomously. The ultimate goal, then, is to build a system that can map as well as the SRI robot, but at a fraction of the cost and require no human input or effort.

1.2 Interface and Project Goals

This thesis discusses the work done while working at Interface, an international manufacturer of carpet tiles. Interface developed the SCOUT flooring system, which consists of Radio Frequency Identifier (RFID) tags at the juncture of tiles. Originally intended for maintenance monitoring by allowing an easy system to count the frequency of vacuumming, the SCOUT tags can be useful for a variety of purposes, two of them being localization and mapping. By acting as fixed landmarks and combined with Simultaneous Localization and Mapping (SLAM), an RFID reader in conjunction with a laser range-finder can map each RFID to a coordinate location based on the distance from walls and objects surrounding it. Once mapped, these RFID tags then represent fixed points on a map, allowing them to act as an alternative to GPS in indoor locations where GPS signals are weak. The existing mapping platform used by Interface consists of a remote control robot, custom developed by

SRI Robotics. It is costly and requires many man-hours to run the robot and subsequently combine the generated maps into a readable format. Only after many mapping runs and time removing inaccurate reads can an accurate map be made. Once done, however, a reader pinging a mapped RFID tag causes an instant feedback of where the reader currently is located according to the map.

Our primary goal for this thesis is to show that a new robot can be developed that not only builds maps autonomously, but does so at a fraction of the cost of the SRI robot. Both of these are requirements for a practical mapping robot. This means that it must be a robotic mapping system that can be run by anyone, not just the developers. It must be accessible, cheap, and robust; requiring little to no supervision. This will be a "Mapper in a Box." Once the robot has completed mapping the environment, it can be used for any variety of tasks, such as guiding visitors, delivering mail, or monitoring security, but all of these will require the building to first be mapped and navigable. To achieve these goals, several autonomous mapping methods are developed for the Robot Operating System (ROS), including an evolved mapping behavior. These are run on a Turtlebot, an open-source robotic platform developed by Willow Garage to be a cheap system to run ROS. Our robot is dubbed A.T.I.C.U.S.S., or Autonomous Turtlebot for Indoor Cartography, Utilizing the SCOUT System. It is also important that A.T.I.C.U.S.S. must map accurately; as accurately as the existing SRI Robot. Without accurate mapping, it does not serve its primary purpose. Therefore our goals will be broken up in the following matter:

1. Add the capability to read RFID tags to the Turtlebot and dynamically associate them with a fixed location. This means adding an RFID reader to the Turtlebot platform as well as write software to integrate RFIDs into the current mapping software integrated with ROS.

2. Add autonomous mapping capabilities to the current ROS methods. This includes several methods, one which will be evolved.

3. Successfully autonomously map an environment using A.T.I.C.U.S.S.

4. Compare different autonomous mapping algorithms. For this project, we will write and test Random, Greedy, and Evolved Behavior-Based Mapping methods. This is in order to see if method has any impact on the accuracy of the mapping system. For this, we hypothesize that the evolved method will perform the best over the other two. It will be judged on speed of mapping, as battery life is an issue, but also on the quality and size of the maps produced. All of these will be compared to the baseline of a manually controlled mapping run.

5. Provide a map as accurate as given by the SRI robot, yet it is done autonomously.

6. Show that RFID tags can improve localization.

There are two main hypothesis for this work, focusing on Goal 1 and Goal 4. Here we predict the learned RFID Localizer will perform better than not using RFIDs at all. For Goal 4 we predict that an Evolved Approach will perform better than a Random or Greedy Mapping Approach, however, possibly not better than a Manual Approach, which will mostly serve as a baseline to compare the other two. This will also more accurately compare whether or not the RFID Localizer improves the robot's map-making abilities or not.

There are three key concepts necessary for A.T.I.C.U.S.S. to map an indoor environment:

- Simultaneous Localization and Mapping, which allows map building. This is not developed in this thesis, merely a requirement.

- Evolved programs to make decisions on when to use RFID tags for localization to aid in developing more accurate maps.

- Evolved exploratory behaviors which analyze the current map using Hough Transforms and allow the robot to autonomously map an environment.

These topics are elaborated in the following sections.

1.3 Simultaneous Localization and Mapping Overview

Simultaneous Localization and Mapping (SLAM) is required by A.T.I.C.U.S.S. in order to create a map of the environment. The previous Interface robot developed by SRI is exclusively used for SLAM. Without using a SLAM method, the RFID tags in the SCOUT Flooring system will not gain location information.

The best way to describe SLAM is maximizing information on location while at the same time expanding the current map. The best algorithms focus both on minimizing the uncertainty of the location as well as the amount of unknown information about the area [63]. Put practically, a map is created of the environment in one pass. This requires that while the map is being made, the location of the mapping unit must be estimated and adjusted, as well as the map it makes. This normally requires several different types of sensors which fit into one of three categories [53]:

- Range-measuring sensors measure distances to objects. This is generally done by sending a signal, be it a light, sonic, or radio-wave based, and measuring the time it takes to get back to the source. Tactile and bump sensors also fit into this category, albeit they only measure a very short range.

- Imaging sensors collect images from the environment in a more passive way than range finders. These can include cameras or stereoscopic cameras.

- Proprioceptive sensors convey information to the robot about it's current state. Odometers, gyrometers, and motor shaft decoders are included in this category. These are useful, but limited in scope and normally require another sensor to aid in mapping.

These types of sensors work best in collaboration, whether they actively measure the environment or passively collect data. Sensors and software work together to allow the platform to estimate it's current location on the known map. This is done while simultaneously building any unexplored area into the map and looking to where the robot can move next.

SLAM is a key area for robotics, as Localization is necessary for any complex, real-world behavior [3, 8, 20, 23, 24, 52, 54, 56, 58, 59, 60, 61]. There are many different techniques for this, but the basic idea is the same for all; the robot must map the environment while navigating through it. Navigation also requires location and pose estimation, or localization. Particle Filters, or specifically, Rao-Blackwellized Particle Filters have been more recently applied to many robotic platforms [3, 8, 13, 22, 23, 40, 52, 58, 59, 62, 72]. These rely on pose estimation and mathematical calculations based on probabilities to determine the robots most likely position. For more information on how this is performed, see [23] or [62]. While this thesis attempted nothing novel in terms of SLAM algorithms, adding RFIDs to current methods improved results using current SLAM techniques.

1.4 RFIDs

Radio Frequency Identifier (RFID)tags are used by Interface in the SCOUT Flor. They allow digital interaction between the carpet tiles and an RFID reader. This allows the carpet itself to provide location information, once it has been mapped using a SLAM algorithm.

RFID tags consist of an antenna and data chip. They come in two general varieties; passive, which have no internal power, and active, which do contain a power source. There are also hybrid chips, which function as passive tags until pinged by a reader, then switch into a high-power mode. For this thesis, we will focus on passive chips, as they are lower cost and small enough to be applied to nearly any item or surface un-noticably.

While they have been around since the 1940s after being invented by Leon Theremin, RFIDs are becoming more and more popular thanks to the proliferation of small computer systems and are primarily used for inventory systems. Most RFID tags are High Frequency (HF) and are used in Near Field Communications (NFC), such as swipe-less credit cards and security door passes. However, there are also Ultra High Frequency (UHF) tags, which provide a greater read range, but require more powerful readers in order to do so.

RFIDs act as an instantly scanable unique identifier, much like how a barcode is used. However, RFIDs offer the advantage of storing more data than a simple barcode or QR Code, as well as being more secure in the case of HF Tags. By having a limited read range and unique IDs, each RFID tag can act as an invisible digital thumbprint, associating each tag with a specific product, person, or device to a computer-based system. This is why it is the obvious choice for security door keys and for Swipe-less credit card readers.

1.5 Genetic Programming

Genetic Programming is used for this thesis as it will provide a near-optimal strategy for reading and mapping RFID tags. Also, by evolving the methods instead of writing them, a better solution can be achieved than by traditional programming techniques.

Genetic Programming is a specialized subset of Genetic Algorithms (GA). GAs are inspired by natural selection. These are used to evolve optimized solutions to problems. The key difference between GAs and GP is that where GAs evolve solutions for problems, GPs evolve programs that are capable of running and completing a task.

There are several main features that define any GA:

- Individual: A proposed solution to the problem.
- Fitness Function: Evaluates the individuals ability to solve the problem. Assigns a numerical value so that individuals can be compared.
- Population: A collection of individuals.
- Selection Process: The method used to select which individuals are paired for mating. This can be random or through a tournament-style process.
- Crossover: Combines two individuals to produce their offspring.

- Mutation: Alters an individual slightly, in order to prevent the algorithm from evolving in only one direction. By continuously trying new approaches, there is a greater chance for optimization.

- Generation: One sequence of taking a population, selecting mates, producing offspring, and mutating them.

In GAs, the primary unit is an individual. This individual is represented by a genome of binary code, integers, floats, strings, or any changeable sequence. This individual represents a solution to the problem. A population is a group of individuals. In the most basic GA, a Fitness Value is assigned to each individual based on a Fitness Function in order to determine which individuals are a better solution to the problem. This function is dependent on the application, but judges the performance of each individual and rates it numerically.

A single generation of a GA begins by evaluating each individual of the population with the Fitness Function. After all the individuals have a fitness, there is mate selection, crossover, and mutation, where both crossover and mutation either occur or do not occur with a certain probability. Two mates are chosen out of the population. There are many ways to select mates, but one of the simplest ways is tournament selection, where several individuals are chosen at random from the population and the one with the best fitness is chosen as a mate. This process is then repeated to choose a second mate, forming a mating pair.

At this point, crossover occurs, where two children are created from the parents by selecting a segment from one parent and one segment from the second parent. These are combined into one child. The second child is constructed from the remaining portions of the parents' genomes. After this, mutation occurs, during which each childs' genes have a probability to be mutated. This mutation is dependent on the format of the genome, but can consist of bit-flips, random numbers, or a unique mutation given the type. After this is done for every individual of the population, a new generation starts. Evolution ends when either a set number of generations is run or the average fitness of the population reaches

a determined limit or plateau. The fittest solution is then left as the chosen algorithm and evolution ends.

Genetic Programs use a similar evolution strategy, the key difference being that the individuals represent programs instead of optimized solutions. Another key difference is that in a GP, the fitness is determined by running the evolved program and determining how well it does the task it is intended to do.

1.6 Hough Transform

Hough Transforms are used by A.T.I.C.U.S.S. in order to analyze the rooms it maps for their basic geometric shape. By analyzing the current map and applying a Hough Transform to it's image, the basic geometric shape of the room can be determined. This allows A.T.I.C.U.S.S. to perform different moves and maneuvers based on the shape of the room, causing it to behave differently if given a hallway or a closet.

Hough Transforms refer to analyzing an image based on its pixels and determining what geometric shapes it contains. This allows for simplifying an image full of data into a more simplified collection of geometric shapes. Put simply, it finds lines based on color differences, generally relying on black and white. The Simplified Hough Transform (SHT) focuses on finding all possible lines a pixel could be contained within in the image. All possible lines are stored in an accumulator array. Each subsequent pixel found on an existing line then increases that line's score. Once all the lines are accumulated, the most viable lines will have the highest scores. A threshold is set to indicate which lines are not likely and therefore discarded once the totaling is complete [30]. This also works with curves in a similar fashion.

Beyond the SHT, there are also the Probabilistic Hough Transform (PHT) and the Randomized Hough Transform (RHT). The PHT is similar to the SHT, except has been shown to use only 2% of the points used by SHT and still retain its accuracy [34, 42]. This allows the computations to be performed much quicker, and analysis of the image to be done nearly

instantly on a powerful enough processor, allowing these processes to be run on a robotic platform in real-time.

RHT was developed by Xu, Oja, and Kultanen [67, 69]. It is the simplest of the three. While the idea behind RHT is similar to SHT, it takes a different approach. The image is broken down into a list of points that fit the color criteria for being considered a line. Two of these points are chosen at random, and the line between them is calculated using:

$$y_1 = ax_1 + b$$

and

$$y_2 = ax_2 + b$$

and the line score starts at 1. This is done until the list of edge points is exhausted. If a new line fits on a preexisting line, its score goes up, if not, a new line is stored. There is also a cutoff for what will be considered a true line, and it is normally small, either 2 or 3 line segments counted. At the end, the true lines are all that are left. The main advantage of RHT over other Hough Transforms, apart from PHT is it is incredibly fast. It is also much simpler than PHT.

1.7 Behavior-Based Robotics

Behavior-Based (BB) robotics is used by this thesis in order to develop unique techniques for mapping based on the shape of the current room. Each room shape can then have it's own behavior for mapping. By having separate behaviors for each, a more optimized approach is possible. BB robotics also adds an additional layer of robustness: by having more versatile yet simple behaviors and combining them based on the stimulus of the environment according to a few simple rules, the robot is less likely to find a situation it cannot deal with. In this thesis, mapping behaviors are evolved by a Genetic Program that allow the robot to autonomously navigate an unknown space robustly.

Behavior-Based robotics is a concept that was originally attributed to Rodney Brooks. In his papers he formulates the idea of a Subsumption Architecture [5]. This describes a completely reactive system, where complex behaviors arise from a few simple programmed reactions to stimuli. This also means the system is more robust, as simplifying the stimulus and behaviors allows them to apply to a greater range of situations.

For a simple example of Subsumption Architecture, we will describe a light following robot consisting of two separate control layers. The first layer will consist of an obstacle avoidance system, where sensors can detect obstacles and commands can be sent to the actuators to enact a move, allowing the robot to avoid obstacles and not caught behind or on them. The second layer will consist of light detection and separate commands to move the robot towards the source of light. With only these two relatively simple layers, a robot can follow a light through a maze or around objects. When more layers are added, the robot can move in ways that would normally be attributed to simple intelligences- robots run from light, or herd around sources of "food"- showing that seemingly complex behaviors can be ascribed to the overlapping of many simple rules and command structures. These types of robots are the best illustrations of Artificial Life, where simple structures allow for seemingly complex, emergent behaviors. This is known as Behavior-Based Robotics.

BB robotics can achieve greater results when combined with Machine Learning techniques, such as Genetic Algorithms [11, 14, 16, 33, 45, 64], Artificial Neural Networks [1, 16], and Fuzzy Logic [28, 39, 49]. More complex, non-behavior-based methods can also be added, allowing for upper level processes like mapping [41, 55]. Most Behavior-based systems use custom software due to the very specific architecture and actions that must exist [16, 27, 48].

1.8 Contributions of this Work

In working on A.T.I.C.U.S.S., we find that RFIDs do add another localization source for SLAM, improving current algorithms, specifically Gmapping, which is used by ROS [23]. We also show that by using Genetic Programming, map quality can be increased by allowing a

decision on whether or not to use a particular RFID tag for localization. This creates a more robust system that can deal with errors, not exclusively relying on one source for location information, but a myriad of sensors. Through this work, Interface also gained a major piece of demonstration hardware that will help aid in the adoption of their new SCOUT Flor System, which is still an abstract concept. With a working robot, early adopters can see the RFID tags in action and perhaps think of their own ideas and programs to run on an RFID embedded floor. A minor contribution of this work was writing autonomous mapping software for ROS, which it was lacking, and needs in order to truly be an open-source system for autonomous robotics. This code, as well as all code written for this thesis, can be found at: `<https://code.google.com/p/aticuss/.`.

1.9 Related Work

This section summarizes important sources that directly influenced the course of this thesis. Some sources are less relevant to the specific problem, yet provided important information. Here the author discusses autonomous mapping, mapping with RFID tags, and Behavior Based / Hybrid architecture mapping robots.

1.9.1 Autonomous Mapping

Mapping environments, indoors as well as out, is a growing field for robotics [3, 8, 15, 20, 23, 24, 52, 54, 56, 58, 59, 61, 60]. Since the rise of GPS and Smart Devices, people, and machines, like to know where they are in the world. Digitizing an office space has many practical applications, whether they be for security purposes, aiding visitors, or for emergencies, such as fires or Search and Rescue operations [8, 20, 23, 40, 55, 58, 61].

Robotic evolution for mapping behaviors, both real-time and in simulation, is becoming more widely used[12, 28, 41, 45, 55, 70]. The approach in this thesis is very similar to Nordin's early work in evolving simple robotic behaviors [45]. In Nordin's paper, a small population

of robot behaviors is evolved in real-time using GPs in order to autonomously map a small, controlled environment.

A good example of evolved behaviors applied to a mapping problem is Schmidt's work on mapping an office [55]. In this paper, the robot autonomously maps an office environment. The behaviors are broken into three levels:

- Obstacle Avoidance
- Local Exploration which uses evolved behaviors to navigate the room the robot is currently in
- Global Exploration, which handles the overall planning of the robot. A list is kept of all currently unexplored locations in order for each to be visited.

Schmidt's work is also a very good example of a hybrid Behavior-Based System. While BB robots can be simultaneously simple and robust, they also lack directive, and wander based on simple rules. This means they do not have any planning, so act less efficiently than a more structured robotic system. A hybrid BB robot uses a combination of a structured planning algorithm and a low level BB system for obstacle avoidance and robustness. Local Exploration consists of behaviors like wall following, random cruising, and movement to the door, which illustrate many common behaviors found in Behavior Based robots. Global Exploration consists of tracking points of interest, similar to Frontier Exploration, where all edges of the currently known map are marked as the Frontier, and kept in a list that is to be explored [70]. This hybrid structure allows for the simplicity of BB robotics, yet the ability to plan and carry out larger tasks that are not found in most purely Behavior-Based robots [12, 28, 41, 55].

In terms of SLAM, ROS utilizes a Gmapping node, which is based on Rao-Blackwellized Particle Filters [23]. Rao-Blackwellized particle filtering uses a cloud of particles, where each particle represents a possible trajectory of the robot. After taking all available sensor readings, such as odometers, laser range-finders, gyroscopes, stereoscopic cameras, the data is

filtered. The key steps to the filtering process are Sampling, Importance Weighting, Resampling, and Map Estimation.

- Sampling generates a set of particles based on the previous generation of particles and the current sensor readings.
- Importance Weighting assigns a value to each particle based on the strength of its claim.
- Resampling redraws particles based on their importance weight.
- Map Estimation estimates a map for each particle. This is based on the current trajectory of the particle and previous observations.

This allows for a map to be constantly re-evaluated as it is being built, providing for a more robust localization scheme.

1.9.2 RFIDs in Exploration and Mapping

Using RFIDs in mapping has become a relatively simple way to digitize information. RFID tags are very good at acting as a static landmark associating a digital value to a real world object. They can also contain information relevant to a human operator, by either storing a small amount of data or providing an link to an external source, if a network connection is available. This has been most notably used in navigating for the blind [37, 66]. This allows a computer to aid someone who is visually impaired and serve as a guide. RFIDs are usually placed on the walls to allow them to be detected from a greater distance [25, 31, 37]. However, RFIDs can also be placed on the ground, as in Kleiner, Prediger,and Nebel's paper as well as the work done by Willis and Helal [35, 66]. In Kleiner, Prediger,and Nebel, a team of robots uses RFIDs as drop-able landmarks [35]. While the tags themselves are not used as landmarks, they do serve as information buoys so that a team of robots, otherwise out of communication from each other, can still build a map together and not re-explore previously explored territory, allowing them to act more efficiently.

In several of these papers, stereo RFID readers have also been used [10, 25, 31]. This allows for triangulation of the tags by measuring single strength variance from a single tag to two RFID readers. This allows a robot to approach a targeted tag using RFID readers alone. In Deyle, Nguyen, Reynolds, and Kemp's work, the tags are not used on the walls, but rather prolifically, on objects and on people [10]. This allows the robot to instantly distinguish tagged objects and tagged people from one another, since each tag is unique. By using two RFID readers on the robot, RFIDs can be located in three dimensions. In most cases, only one antenna simply allows the detection of an RFID tag. By using two antennas, the RFID can be located in space and navigated to through triangulation. In this liberal application of RFID tags, the authors are able to capitalize on the fact that many items already are labeled with RFID tags. Their robot was able to use this information to pick up certain objects and deliver them to certain people [10].

1.9.3 ROS and Kinect in Mapping

Since both the ROS and Kinect have only been around a few years, there is very little literature on this topic. However, Zaman, Slany, and Steinbauer's paper covers major testing using ROS for mapping, using the same software as this thesis [72] . This work shows that ROS can be successfully used for mapping and navigating a real environment. This will be gone into in more detail later, in the review of ROS.

In Ganganath and Leung, a Kinect sensor is used in place of more expensive laser range finders [22]. The Kinect and odometers on board the robot are combined with the extended Kalman filter (EKF) and particle filter(PF) to provide accurate localization. Here, the Kinect is accurately used in conjunction with Hough Transforms to detect landmarks, which are a key tool for localization. Through experimentation, Ganganath and Leung show that the Kinect can be used to accurately aid in odemeter-based localization through landmark detection [22]. Fallon, Johannsson, and Leonard also show that the Kinect can be used for accurate localization [13]. In a similar technique to Ganganath and Leung, the Kinect is combined

with PF, also known as a Sequential Monte-Carlo. The Kinect was used without any other form of localization sensor, such as odometers or gyrometers, and tested on a variety of platforms, such as a robotic wheelchair, a PR2 from Willow Garage, a quadrotor helicopter, and a wearable mount. For more information on the Kinect sensor, EKF, and PF, please refer to Fallon, Johannsson, and Leonard or Ganganath and Leung [13, 22].

However, none of these papers use the Turtlebot, which is a much cheaper platform than either of the robotic systems used in either Zaman, Slany, and Steinbauer or Ganganath and Leung [72, 22].

1.10 SCOUT and A.T.I.C.U.S.S.

In this chapter we introduce the SCOUT Flooring system developed by Interface as well as go into detail on the two robots, the SRI robot and ATICUSS, used to read and utilize the RFID tags in the SCOUT Floor. The first robot was developed by SRI for Interface, and the second robot, A.T.I.C.U.S.S., is the work of the author.

1.10.1 SCOUT Floor and Interface

Interface is one of the world leaders in carpet tile manufacturing. The company is attempting to reach a waste-free manufacturing system by 2020. Currently, scrap from the production process and old tiles are being used to create recycled carpet. Further reduction in waste for the entire manufacturing process is also being researched and developed, as well as moving towards more natural bases to create their carpeting.

Interface uses Tactiles, a square adhesive to attach four carpet tiles together at their corners. This allows for quicker installation and nearly instant removal. It also allows for less waste than carpet rolls or carpet tiles that use glue. Recently, Interface began attaching RFID tags to Tactiles, dubbed SCOUT Flooring. See Figure 1.1 for an illustration of where the RFID tags lie in the SCOUT system. Originally, SCOUT was developed for maintenance monitoring, so each tag could act as a counter for how many times a carpet tile was vacuumed.

This is for quality assurance as Interface warranties its carpet for a certain number of years as long as it is properly maintained. However, ideas quickly spread since as soon as carpet became able to be identified digitally, many options were possible. The carpet could be used as a security system, as an interactive tool or display, could provide links to maps or information about the current room. One of these was mapping, and using each tag as a location in space, in order to track any device equipped with an RFID reader. Once a device reads a tag, that tag corresponds to a location in space. If used with a network, a centralized system would know the location of every person or device equipped with an RFID reader. This provides an alternative to GPS, which does not work in some buildings due to materials or the number of floors above. It also serves as an alternative to odometry or gyro based sensors, both of which have a percentage of error that accrues over time. However, this first requires the tags to be mapped to their environment [71]. Once each tag is placed, it must be read by a mapping robot equipped with a range-finder. After multiple reads and ranges, the distance from each visible wall will be associated to the tag. Once this process is complete, the tag has a location with respect to a map of the room.

1.10.2 The Old Robot

In order to utilize the SCOUT Flooring system, Interface hired SRI to develop a remote-control mapping robot. Based on the Segway, SRI developed the mapping software as well as integrated an RFID reader. See Figure 1.2.

This robot can make accurate maps, as can be seen in Figure 1.3. This is Interface's Aware House, which serves as an office and showroom for their sales representatives and potential clients. It has tags attached to the carpet, both HF and UHF. However, adding the RFID's identifier string to the map requires them to be sequentially numbered in a Cartesian coordinate plane as well as manually combining multiple map runs. This means each tag must be manually numbered according to it's location in the room. For example,

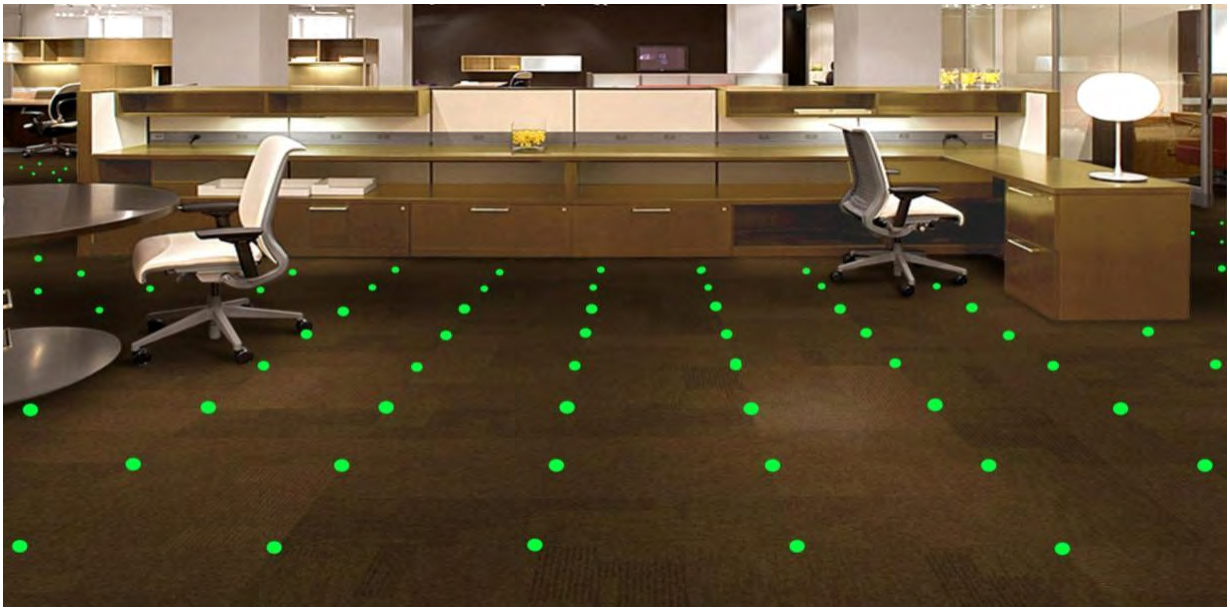

Figure 1.1: The SCOUT Floor

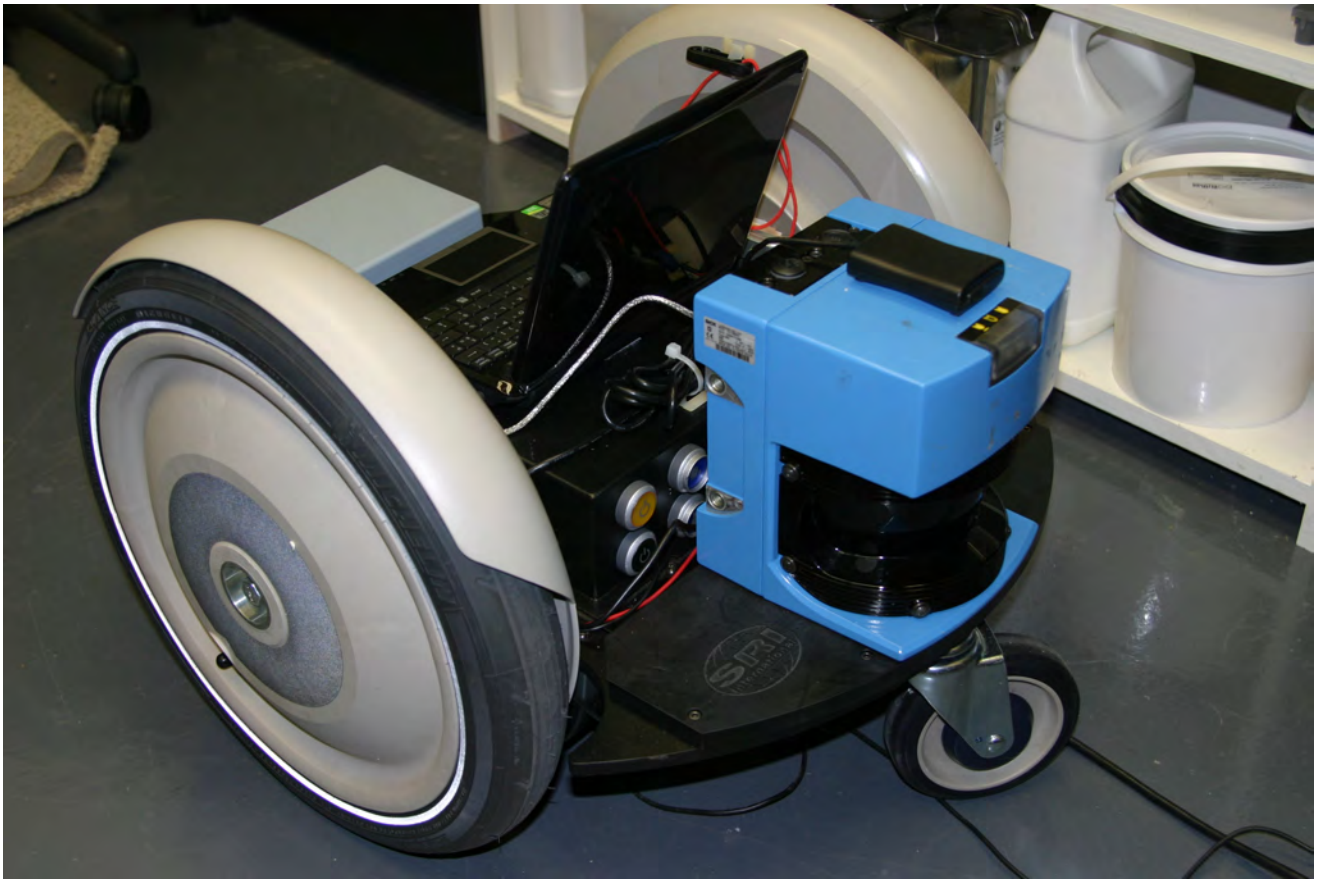

Figure 1.2: SRI Mapping Robot

if starting in the Northwest corner, the tag will be numbered 0000. The tag to it's East will be numbered 0001 and the tag to its South will be numbered 0100.

For example:

0000	0001	0002
0100	0101	0102
0200	0201	0202

This must be done for every tag in order to aid the localization software. Inaccurate reads are discarded manually and all other reads for the same tag are summed and averaged for a more accurate location. This location is then lined up to a grid that corresponds to the RFIDs number string. Our goal for this thesis was to show that this could be done autonomously and in one run, as well as much cheaper. While this was not achieved, shorter runs were able to map a segment of the building. The search for a cheap robotic platform lead us to the Turtlebot.

1.10.3 Turtlebot

Turtlebot is an Open Source platform developed by Willow Garage [40]. It runs the Robot Operating System (ROS), which is also Open Source and developed by Willow Garage. The robot's hardware consists of the Create developed by iRobot, (started by Rodney Brooks), a control laptop, and the Kinect Sensor, developed by Microsoft originally as a game controller. The Create serves as the base of the robot and the locomotion, the laptop serves as the controller, and the Kinect handles vision. All of these components are open source. In order to read RFID tags, an RFID reader was added to the Turtlebot.

1.10.4 ROS

The Robot Operating System (ROS) was developed by Willow Garage for robotics applications and is based on Ubuntu Linux. It can run on almost any platform, ranging from the Turtlebot to the human-sized PR2 humanoid [40, 50]. ROS consists of packages, some of

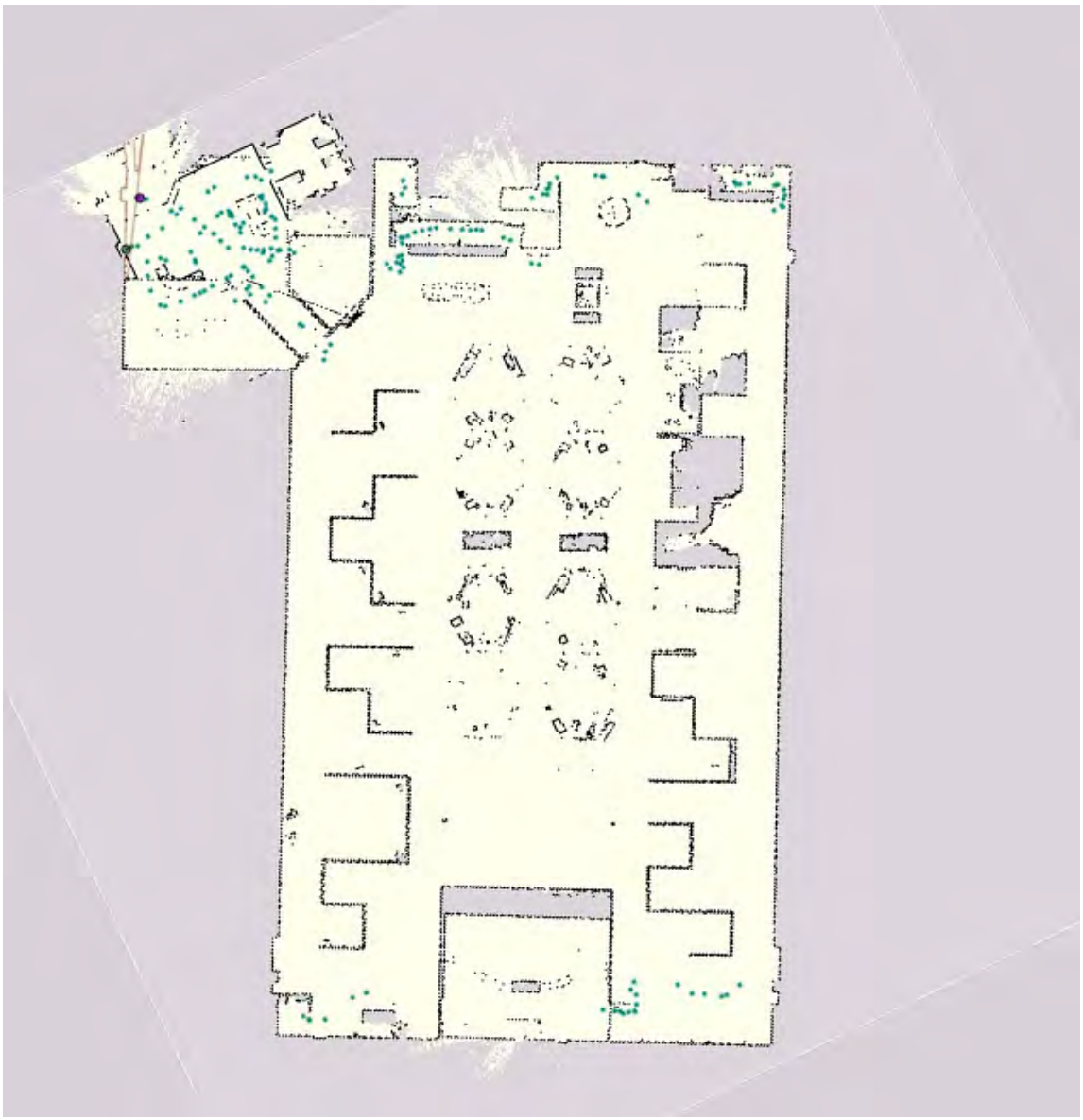

Figure 1.3: Map of the Aware House, mapped by the SRI Robot

Figure 1.4: A.T.I.C.U.S.S.

which are developed by Willow Garage, but most are developed by the ROS Community. For the Turtlebot itself, many packages come with the ROS install, including autonomous navigation of a known map, a SLAM algorithm based on gmapping [23], the ability to follow a person, and remote control functionality. Other packages for ROS consist of voice-control, text to speech, and integration for multiple robots.

Thanks to a large community and the backing of Willow Garage, there are many Wikis, forums, and tutorials available. More importantly, ROS comes with RVIZ, a visualization tool that is immensely helpful in debugging. Not only does it visualize the robot and its environment, it can also directly display readings from sensors, such as what the robot is currently seeing or where the location of the robot is according to the odometers. It is also easily customizable. See Figure 1.5 for a display. Here we were able to add graphics representing the RFID tags as they were read during mapping.

The ROS architecture is built around nodes. Each node is an individual program. Nodes can subscribe or publish to any other node as topics, which is the primary form of communication across nodes. Nodes publish topics and messages, which are simple message packets that can be passed between any node on a computer to any other node, as a one-to-many relationship. These messages can also be passed over the network, so nodes on separate machines can subscribe to topics being published on a remote machine, such as the one controlling the robot. This allows for simple nodes to be written that are easy to understand, yet interact with the robot in an variety of different ways by customizing messages sent between nodes, and having nodes that interact with various sensors or actuators. It also allows for easy integration of any new node into the existing architecture. This allowed a relatively simple process in order to create the autonomous mapping robot, A.T.I.C.U.S.S.

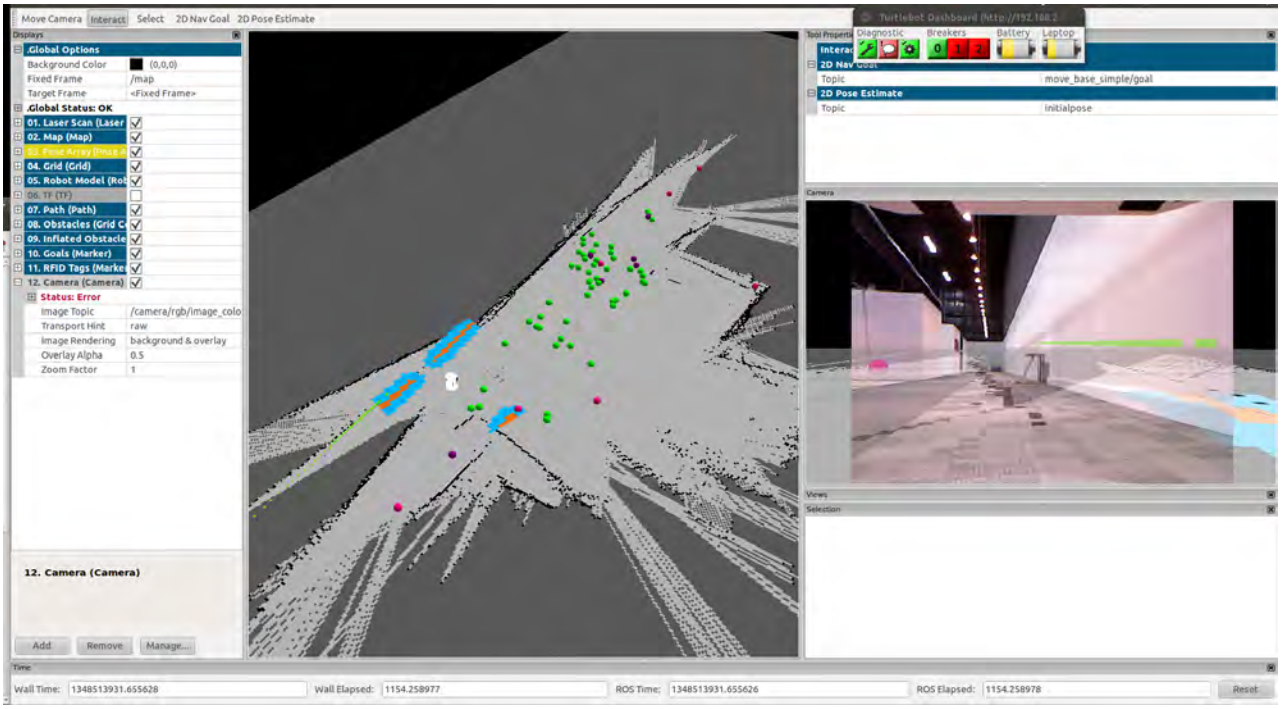

Figure 1.5: RVIZ During one of A.T.I.C.U.S.S.'s runs. The Green Dots are RFID tags. On the right is a feed from the Kinect Camera onboard.

Chapter 2

Methods

This work was done while working at an Internship at Interface in the Summer of 2012. This means that time with access to the correct resources was limited, primarily the Aware House, which has already been outfitted with the necessary RFID tag distribution. This limitation effected how long the robot could train exploratory behaviors. It also affected the sample size of produced maps under each mapping algorithm. The best of each is compared in the Experiments section.

2.1 Hardware and Software Setup

The hardware can be broken into three categories:

- The Robot: Turtlebot, as described above. Consists of a Create, laptop for processing, a Kinect Sensor, and an RFID reader.

- The Environment: This is the Aware House, the demonstration building for Interface's innovative and high end products. Since it has already been configured for mapping, RFID tags are placed at the junction of every carpet tile. This is necessary to test how much a distributed RFID tag system aids in localization.

- Control Computer: This is used to interface to the robot. It runs Ubuntu Linux with the ROS SDK, necessary for connecting to and controlling the Turtlebot. From here, commands can be sent to the Turtlebot over Wi-Fi, as well as the feed from the Turtlebot be displayed. RVIZ runs on this machine, as well. It also serves as the controller when A.T.I.C.U.S.S. is controlled manually.

This paper uses ROS version Electric for running the Turtlebot. All code was run on this operating system.

2.2 Necessary Steps for the Development of A.T.I.C.U.S.S.

Most of the time while working at Interface was spent getting various kinks out of the robot-control computer system as well as learning Python and the intricacies of ROS. Once the turtlebot could reliably be accessed, it was run many times, both autonomously and via remote in order to ensure that it could be run unsupervised for any period of time. This included refining the collision bubble around the turtlebot: a virtual threshold that prevents the robot from getting too close to an object and getting stuck.

Once the robot could be reliably run and avoid colliding with obstacles it couldn't see due to the placement of the Kinect, the RFID Reader had to be trained and the various algorithms for mapping had to be developed. These algorithms are as followed: Random, Greedy, and an Evolved Behavior Based Method, with a remote run as a control. Once this was done, they could be tested head-to-head by comparing their mapping results. Each method was started at the same location and time was recorded, limited to how long the robot could map until the map began degrading. This symptom was discovered during the collision testing phase. As the battery gets lower, the sensors become more inaccurate and eventually begin over-writing any good mapping results with poorer ones.

Chapter 3

Development of A.T.I.C.U.S.S.

In this chapter we describe the necessary programs added by the author to ROS in order to allow A.T.I.C.U.S.S. to autonomously navigate and map an environment fitted with RFID tags. This approach focuses on combining techniques covered in the Related Work Section. While some methods were specifically written for the topic of this thesis, like RFID Mapper, many programs in ROS were simply adapted, such as the controller MoveBase. The first section covers the evolution of the Genetic Program RFID Mapper, which is the decision engine for whether or not to use an RFID tag for localization. In the Autonomous Exploration section, each exploration behavior developed is described. These are Random, Greedy, and the second GP, Behavior Based Exploration.

3.1 Nodes Pre-built into ROS used by A.T.I.C.U.S.S.

ROS has all the complexity and power of a desktop operating system. This means that much of the architecture and structure of the robot is completely taken care of by the distribution of ROS for the Turtlebot. There are also many additional nodes that can be used that have been written by Willow-Garage or the ROS Community that add greater functionality to the robot. However, dependent on how specific the task is, more than likely some nodes will have to be written by the user to accomplish their individual goals. These nodes are explored in full detail in [72].

There were several nodes built into ROS that were left unmodified for this thesis yet allowed integration into autonomous RFID mapping. These nodes covered the sensors and

movement of the robot. The decision processes had to be written and integrated with these nodes:

- Gmapping is the SLAM node that comes with the standard Turtlebot distribution of ROS. It is based on Grisetti, Stanchniss, and Burgard. [23]. In ROS, it allows the Turtlebot to map an environment while under remote control through a wireless network.

- MoveBase was also used. This serves as the core of the command structure to move the robot autonomously, given it's current location and a destination goal. It also handles obstacle avoidance and path planing. These are both separate aspects of the same control node. The first is a Global Planner, which uses an A* search, used in Marder-Eppstein, Berger, Foote, Gerkey, and Konolige's work developing ROS [40]. The second is local avoidance, which avoids obstacles. This uses a dynamic window approach proposed by Fox, Burgard, and Thrun [21].

- AMCL uses Adaptive Monte Carlo Localization in order to localize the robot. It uses the results of the laser scan, odometers, and gyrometer to estimate the robot's pose. This is a similar algorithm to Robust Monte Carlo Localization, by Thrun, Fox, Burgard, and Dellaert [62].

- In addition to these, the standard sensor nodes were used, unmodified. This allows the use of the Kinect Sensor, Odometers, and Gyros. These as well as Move Base above are part of the Turtlebot Bringup Package.

- There is also an additional RFID reader node which was simply adapted. This was written by Travis Deyle, who wrote the drivers for the RFID reader [10].

For a more detailed description of these nodes used, please see the paper written by Zaman, Slany, and Steinbauer describing in detail the developments and issues in mapping in ROS [72].

3.2 RFID Mapper

Since the robot could already move and sense its environment, adding an RFID sensor and software to integrate this information into the additional localization structure had to be written first. Once this was done, autonomous mapping software could be written.

This section covers the first experimentation with the RFID Mapper software that was written by the author. This allows the RFID tags to be read by the robot as it drives over them on the carpet.

In order to map tags to a location, the robot must read the tag and assign it to a Cartesian grid with respect to its origin. The origin is 0,0 and is simply assigned as the robot's starting location. Each point represents a tile juncture, with the tiles being 50cm by 50cm. There were two important factors that we considered necessary for a RFID Mapping Robot:

- The tags would have random IDs, and their values would not be altered. Having to lay out specific tags to specific locations, such as sequentially numbering them in a grid, as is required for the SRI robot, takes away the autonomy and ease of the system.
- The ability to remap tags. If a tag is assigned an inaccurate location, the robot will be able to reassign its location.

In order to remap a tag, the robot must have qualifications on when to remap and when to use the location given by the tag. For this, two values were chosen.

- Confidence. This is a value used by the Turtlebot. It assigns a numerical value between 0 and 200 to how well the current laser scan of the environment matches the the most recent map. High values indicate a high confidence in the robot's location. This information is based on the covariance of the current map and the current pose.
- Disparity. This is the difference between where the robot believes it is and where the current RFID tag being read says it is. This is a distance in meters.

If the confidence and the disparity are higher than a certain value, then the tag is remapped, meaning a different X,Y coordinate is assigned to the ID of the current RFID tag. However, both the coordinates and the tag IDs are unknown at start up and must first be discovered before any mapping can be done.

In order to determine how to map RFID tags, we chose to use a simple Genetic Program. The only concern of this program is whether to reassign a current RFID tag to the location of the robot or to trust the tag and use its position as the robot's. It consisted of a small population of 50 and ran until convergence. In the initial population, each individual consists of a random confidence between 0 and 200 and a random disparity between 0 and 5, representing meters. Evolution of the GP was done in real time on the robot, and every tag read caused an evaluation of the individual selected for testing in the current generation's population. Therefore, once a tag is detected for the first time, that tag's ID is assigned the current X,Y position of the robot. Every subsequent read causes a decision to be made; whether to publish the location of the tag to the SLAM Gmapping Node, or decide that the location is inaccurate and remap the tag.

The RFID tags in the Aware House are sequentially numbered in order to be read as rows and columns. This was done in order for the tag locations to be adjusted manually after mapping with the SRI robot. This requires, however, that each tag be manually and deliberately written to once placed on the floor. This also means that if an RFID tag stops working or is defective, it must be replaced and manually numbered appropriately. We made a decision not to use the fact that adjacent RFID tags would be similarly numbered in our own mapping experiments because it requires even more effort from the user as it requires each tag to be manually numbered. It also adds complexity and detracts from the robustness of the robot: if a tag were mis-numbered or missing, the robot would loose a localization source. The proposed system uses a purely random RFID grid, where the only preparation needed is to install the carpet tiles using the SCOUT tags as opposed to the normal Tactiles. In this way, it requires no additional effort to install before mapping is done. However, a

sequentially numbered system is a good resource for training the confidence and disparity values, so this was used during the evolution of the RFID Mapper GP. However, for all subsequent evolution and experiments, this function was turned off.

Being able to recognize RFID neighbors by ID alone allowed the RFID GP to gauge whether or not an individual consisted of a good confidence and distance values. Since this is a GP and evaluation is done after performance, fitness values were not used; individuals were either mutated if they performed poorly or procreated if they did well. This was based on the sequentially numbered tags. This being the case, this is technically not a GP, but merely a simplified algorithm, a Survival of the Fittest Algorithm.

With a sequential tag system in place, RFID neighbors are easily identifiable. If the current RFID tags location is farther than one meter from its nearest neighbor, it was mapped poorly and needs to be remapped. However, if it is close to its neighbors, then it was mapped well.

If the individual gave a poor indication on whether to remap if above the confidence and distance thresholds, it was mutated to greater numbers, causing remapping to occur less often. If an individual indicated to use the tag location when it was mapped poorly, the thresholds were lowered to cause remapping to occur more frequently. If it did well, it was duplicated and replaced a random individual in the population. As soon as the decision is made and the individual is treated accordingly, the next individual in the population was chosen and used to decide the outcome of the next RFID read. After a few thousand reads there was convergence when all the individuals in the population had the same values for disparity and confidence. These values were then used as the confidence and disparity for all experiments performed below that utilized the RFID tags. Also, once the RFID Mapper had been trained, all exploration techniques were trained without the ability to recognize if RFID tags were neighbors by ID alone. That is, all RFID IDs were taken as random and arbitrary.

3.3 Autonomous Exploration

Once A.T.I.C.U.S.S. could successfully detect, distinguish, and store separate RFID tags, the autonomous mapping methods could be written.

Since ROS does not have an autonomous mapper program, one had to be written, or rather, three were written. Since it does have an autonomous navigation ability which relies on the package known as Move Base, this was used and adapted into three autonomous methods for exploring an unknown environment.

The following sections describe the various autonomous exploration algorithms developed in order to test the above RFID Mapper. In order to understand the architecture of the robot during the experiments, see Figure 3.1.

3.3.1 Random

The first method was Random Mapping. This method moves the robot to a random location of the already known map. While counter-intuitive, as the robot can only move to positions on the already existing map, by moving to a new location, more area for exploration is revealed. This means the most efficient moves for this method are going to be on the edge of the known map, however, the pattern is still randomized. A random move into the middle of the known map will add very little new information, if any. See Figure 3.2.

3.3.2 Greedy

The second algorithm developed was Greedy Mapping. This takes a greedy approach, where it moves to the closest location with the most amount of unknown area. It is inspired in concept to a Frontier Approach [70], however, involves less planning. Frontier Approach remembers unexplored areas and marks them as frontiers. These are added to a stack and eventually, all frontiers are explored. Greedy is similar to this in that the most unexplored areas are generally found at the frontiers of the map. The rest of the algorithm is relatively simple: ten random points are chosen on the currently explored map, and each area within

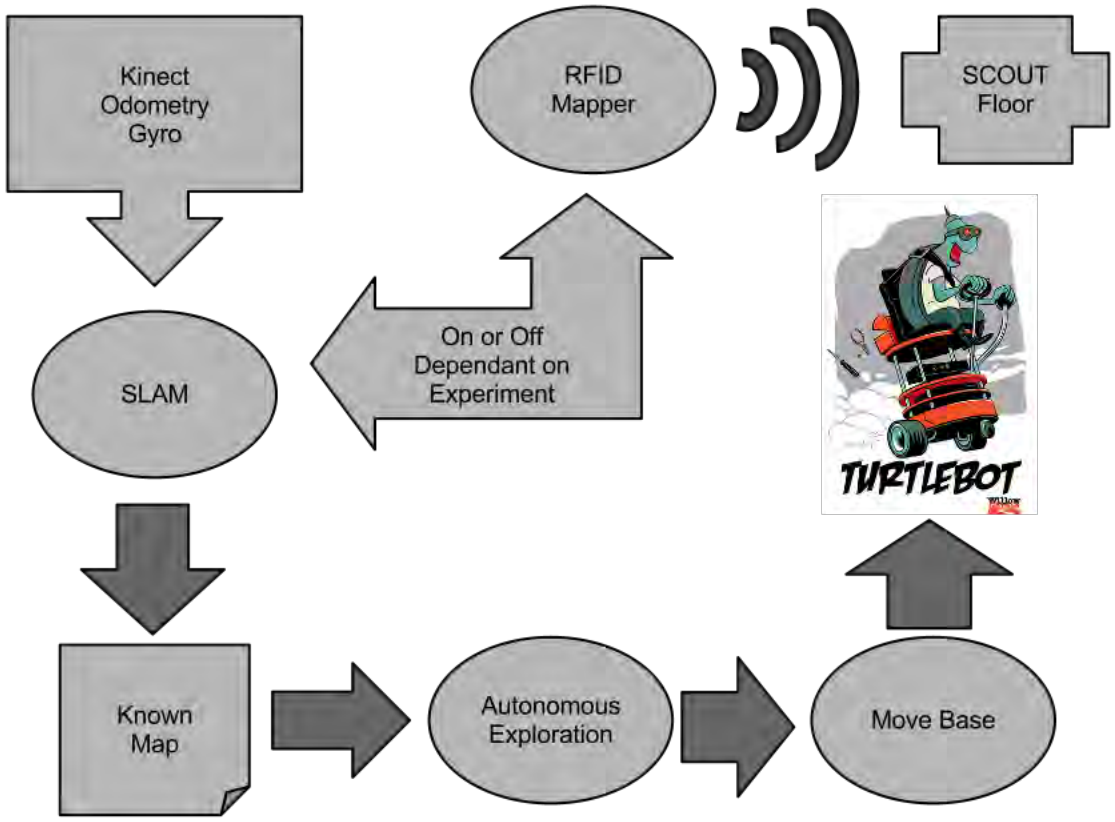

Figure 3.1: Architecture of A.T.I.C.U.S.S. Here, the ability to read RFIDs can be turned on or off. When on, it sends a location signal to the SLAM node. The SLAM node is also constantly receiving data from the Kinect Sensor, the odometers, and the Gyro. The SLAM node, in turn, verifies the location and expands the current map. The autonomous exploration method then reads this map in order to determine the next move. Once a decision has been made, a command is sent to Move Base to move the robot to a new location.

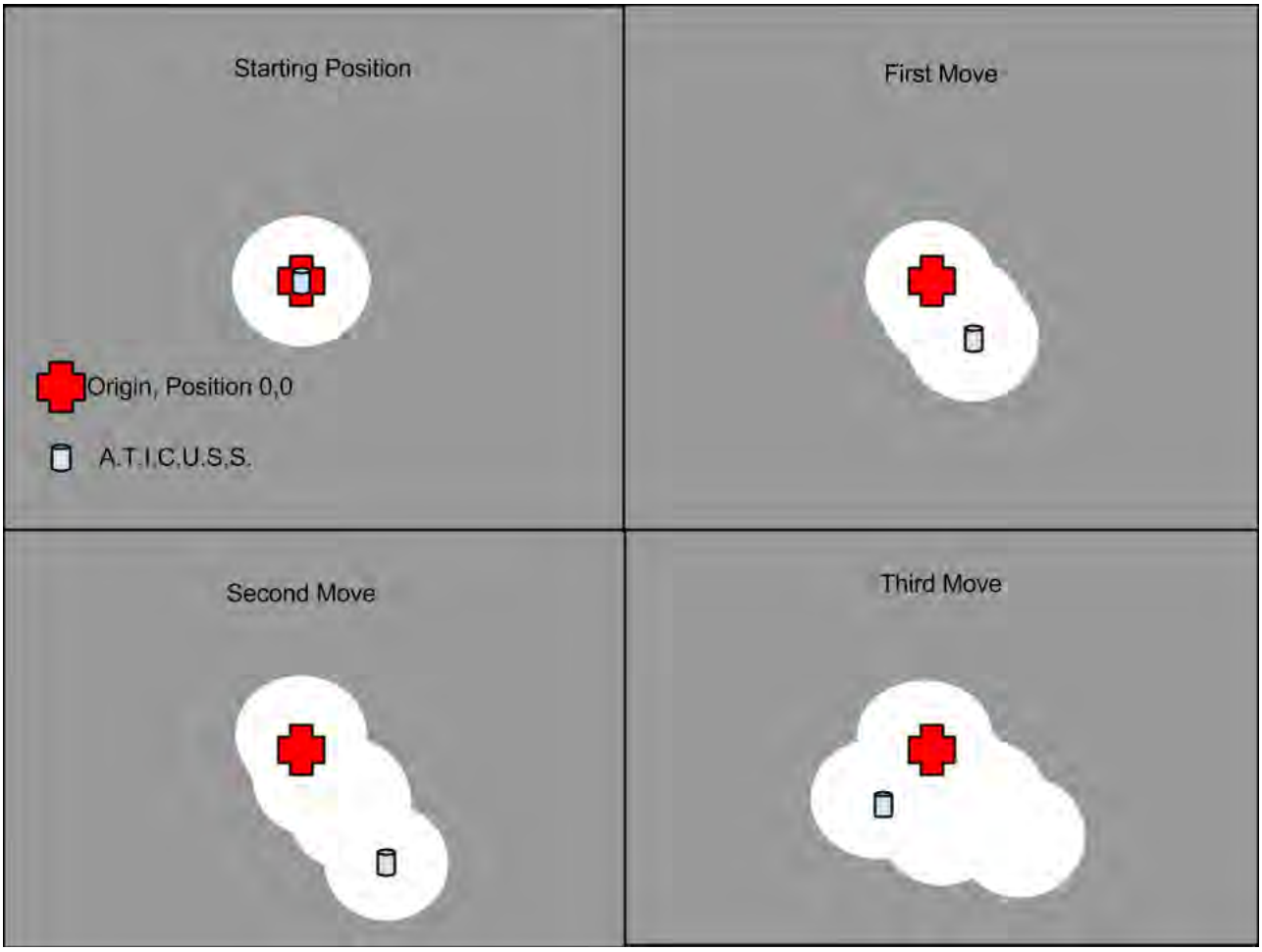

Figure 3.2: Explanation of Exploration: white is the known map, grey is unexplored space. At the end of the third move, the robot's current map is larger than the start. It can now move to any area in white and further expand the map.

five meters of each point was analyzed for the amount of unknown space (u). In Gmapping, gray pixels represent unknown space on the map, white represents open space, and black represents occupied space, or rather, walls an obstacles. The area with the most amount of gray pixels is the least explored. The distance (d) to the randomly chosen point was also measured. The point with the highest unknown space to distance ratio:

$$\frac{u}{d}$$

was chosen as the winner and next goal for the robot.

3.3.3 Evolved Exploration

The third exploration method was an Evolved Behavior-Based Exploration. To handle the structure of the evolution, we used DEAP (Distributed Evolutionary Algorithms in Python) [9]. DEAP is a package for Python that allows quick implementation as the only code that needs to be written for the simplest GA is a Fitness Function and the structure of an Individual.

The approach taken for this thesis is not a traditional Behavior Based approach; since Move Base controls the minor navigational aspects of the robot, such as path planning and obstacle avoidance, we did not feel that developing simple behaviors was warranted. So instead, the behaviors dictate where the robot moves next, or more specifically, which algorithm chooses its next move. These moves either belong to a Global Exploration Strategy or a Local Exploration Strategy, much as Schmidt did in [55]. With this being the case, each individual consists of eight floating point values. The first four represent the probability that a certain global exploratory move is chosen. The moves are as follows:

- Random: performs a random move, taken from the above Random Exploration Method.
- Greedy: performs a greedy move, also taken from the above Greedy Exploration Method.

- Ten meters forward: moves the robot to a point ten meters forward. If the point is on the other side of a wall, it will try to get to that location. If it can't move to that position, it will get as close as possible.
- No Move. Allows the robot to continue Local Exploration.

These represent a Global Exploration strategy.

Once the robot reaches its next location, we wanted the robot to be able to do local exploratory moves, local being defined as the five meters around the robot after reaching its destination. Each individual also has a custom move list of five moves. These are unique to the individual for each situation, and they are randomly chosen from the following simple moves:

- Face left
- Face right
- Random turn
- Face nearest wall: this is a method adapted from the calibration methods that came with ROS. It simply turns until the robot is facing a wall.
- Move forward two meters.
- Move forward a random distance between zero and five meters.
- Move to local frontier, being defined as an area with unexplored area within five meters.
- Random local move: Moves the robot to a random point in the five meter area surrounding it.
- No Move. This allows the robot to skip local exploration and continue exploring the larger environment.

These custom moves are chosen with regard to an evolved probability, similar to the Global Exploration segment of the genome. The probability that a certain move will be chosen at random is what is represented in the genome of the individual. A high probability of a certain move means the individual will perform this move more often, however, it is not the only move it will perform. This was done in order to prevent stagnation as well as keep the robot more robust for unknown situations. The Local Exploration moves are meant to allow the robot room to evolve different patterns of movement dependent on the configuration of the local environment.

In order to determine the local environment, we used a Randomized Hough Transform (RHT) [67, 69]. It was very simplified, as it was only looking for two walls on the already explored map and comparing them. Since the Aware House is all open space, corners, or hallways, we decided finding two walls was sufficient. This was done by applying a RHT to the local map and adding an allowance to consider a group of similar lines as the same line. If there were no lines, this was considered an open area. If there was one, this was a wall. If two lines were found, the angle between them was found; if they were parallel, it was a hallway, and if they were perpendicular, it was a corner. However, due to the large amount of noise in the map, each of the four conditions got a vote. These votes were adjusted by a modifier, found in the last four floats of the individual. Dependent on the vote and the modifier, one of four custom movesets was chosen.

The Genome Format of the individual is in the following format:

Or rather, a list, such as this:

```
Percentage for: [Random|Greedy|10m in front|Area around Robot]
[Outer Wall (1 line)| Hallway (2 parallel lines) |
Corner (other lines of significant length) | Open/Muddled/Free Space ]
```

Each individual then had four sets of custom moves, one for each area type. Each custom moveset was five moves, one of eight primitive behaviors to be performed sequentially. This

Table 3.1: Genome Configuration for Evolved Mapper

Global Search to Perform
Random
Greedy
10m in Front
Area Around Robot
Types of Wall Configuration
Outer Wall
Hallway
Corner
Open/Muddled/Free Space

was done since simple obstacle avoidance behaviors were not developed above. These were stored separately than the genome, yet still mutated under the same probability.

Table 3.2: Possible Moves for Local Exploration

Integer Representing Custom Move	Custom Move
0	Forward Random Distance
1	Forward 2 Meters
2	Random Turn
3	Right Turn
4	Left Turn
5	Align to Wall
6	Move to Local Frontier
7	Random Move within 5 Meter Area
8	No Move

Therefore, a custom moveset of

```
[0 | 6 | 5 | 3 | 1 ]
```

will move forward a random distance, move to a local frontier, turn to face a wall, turn right, then move forward two meters. There is one custom moveset for each of the four area types, be it outer wall, hallway, corner, or open space.

An additional feature of the custom moves list was a Tie-Crossover. If two votes for the area shape determined by the RHT tied, an in-run crossover of the custom moves occurred. This was done in a method similar to the crossover function; by randomly selecting a mid-point in the move list, performing the first voted custom moveset until that point, then finishing with the last section of the second move list. Once the moveset was performed, the Tie-Crossover was lost and not stored in the individual. This allows for more varied movements in uncertain terrain.

Fitness Function

The fitness of the individuals consisted of the robot's average velocity plus the change of entropy, plus any change in the map size during its run:

$$fitness = \bar{vel} * 1000 + \Delta entropy * 1000 + \Delta map/10$$

The entropy value is calculated by the Gmapping program. It is an estimate of the entropy in the distribution of the robot's position, where higher values indicate greater uncertainty. The velocity was used because of the constraint in time on the run due to the short battery life. A moving robot will always be exploring more area than a robot that is stationary. Velocity allows distance traveled to be beneficial and time spent calculating the next move to be detrimental.

The values of average velocity of an individual and change in entropy are multiplied by 1000 simply because they were incredibly small and we wanted them to be the primary motivation for evolution.

The robot's GP was evolved in the environment; not in simulation. Due to the lack of battery power on A.T.I.C.U.S.S., this was limited to a population of 6 for 14 generations, or four runs of the robot due to limitations in the battery. Each individual in the generation got one move, which determined its fitness.

Best Individual after GP Evolution

After evolution the best individual's genome from the GP was as follows:

Table 3.3: Best Evolved Mapper

Global Search to Perform	Probability of Search Occurring
Random	0.088214122273091888
Greedy	0.94566893173838107
10m in Front	0.87662834339175355
Area Around Robot	0.74001467048082403
Types of Wall Configuration	**Modifying Weights to Voting Power**
Outer Wall	0.57497086623511817
Hallway	0.2077178029975677
Corner	0.83682493551525061
Open/Muddled/Free Space	0.6458008944

Table 3.4: Custom Moves for Best Individual

Outer Wall	Hallway
No Move	Random 5m Move
No Move	Random 5m Move
Random 5m Move	Left Turn
Move to Local Frontier	Random 5m Move
Random Turn	Move to Local Frontier
Corner	**Open/Muddled/Free Space**
Random 5m Move	Random Turn
Forward 2m	No Move
Forward Random Distance	No Move
Random 5m Move	Forward 2m
Random 5m Move	Random 5m Move

This shows that Greedy Mapping was chosen the most, followed by a move ten meters forward and exploring the local area. Random moves were nearly never chosen. In terms of weights for the Hough Transform, they were skewed in favor of corners, meaning that corner moves were chosen the most, which consisted primarily of local random moves. Corner moves

were probably weighted more heavily due to the fact that the area the robot was trained in was primarily corners and walls, with very little straight hallways. In fact, most custom moves were local random moves, with no move being the second most popular option. Aligning the robot to the wall and turn right were entirely removed from the robot's repertoire of local moves.

Chapter 4

Experiments

The best way to test how each mapping program performs, as well as the RFID mapper, is to run them and compare the maps.

4.1 Methodology

A.T.I.C.U.S.S. was tested in two different ways, once it was trained in the Aware house. Firstly, maps of the Aware House were made using the different mapping methods. It was then tested in a second environment, referred to here as the A-smart-ment. Here it was run only in manual mode while the confidence values and disparity vales were logged. This was done to compare quantitatively the differences between localizing with using RFID tags and without.

4.1.1 Aware House

This is the showroom for Interface. It is also a working environment. It has a square footage of 22,895 square feet, so it is a rather large area. This limits the amount of space that can be mapped. With this being the case, the area focused on for mapping is shown in Figure 4.1. This a zoom of Figure 1.3, which was mapped by the SRI robot.

For each of the three mapping algorithms, the best of two options is compared, once with the RFID mapper on and once with it off. There is also a manual run, via remote control, to serve as a baseline.

The major limiting factor on the robot is the battery life. Very early we knew it would never map the entire Aware House in one run, therefore, we started each in the same location.

Figure 4.1: Zoom of the corner focused on by A.T.I.C.U.S.S., mapped by the SRI Robot

Also, because of the limited battery life and a limit on the amount of access to the Aware House, the best of each run is compared. Once the development phase was finished, less than a month remained of access to the Aware House, and each run took nearly a day. While many more maps were generated for each method, most were unusable. This further challenges Goal 3; generating usable maps in one run. This and similar issues will be addressed in the Discussion section.

All runs started in the same location. Originally, we planned on giving each run a set number of moves, but time became a better comparison factor of the methods, as well as subjectively comparing the maps created. This was due to an inevitable degradation of the map based on accrued errors. Whichever method could last longer while still creating a semi-accurate map should be considered better. This also means that better methods produce a larger map, as more area is explored. However, the biggest indicator for how good a method was is to view the map created since the goal is to create usable maps. If one method produces a more accurate map, it is superior.

Times are also recorded. Each run was stopped once the map devolved to a point where it was unusable. While this isn't really a factor in considering which method is the best, the better methods do preserve the map for longer. Ideally, this would not be a concern. Gmapping should allow the mapper to run indefinitely without destruction of the map. However, due to inaccurate sensors and the environment having similar traits repeated throughout, the Gmapping algorithm can get confused and lose the location of the robot, destroying the previous map with the new one. This is why the runs with the RFID reader turned on last longer.

4.1.2 A-smart-ment

This is a much smaller area than the Aware House, 655 square feet, so served better for testing the differences between using the RFID tags or not as the robot was able to make a complete map in one run. The shorter distance also allowed the robot greater visibility

of the walls, which are its primary landmarks. A map of the A-smart-ment can bee seen in Figure 4.2.

Much like the Awarehouse, the A-smart-ment serves as a showcase for potential technology. Like the Awarehouse, it has RFID tags at the junctures of carpet tiles. It also houses pressure sensors for movement and fall detection as well as motion activated lighting. These exhibit possible technologies to help the elderly.

Here runs were done only in manual mode. 10 runs were done for each method; using RFIDs or not using the RFIDs. Each run was timed to 15 minutes. During this time, the Confidence values were recorded every few seconds for both methods, and for the RFID Mapping method, the disparity between RFID location and robot location was also recorded. This data could then be used to quantitatively compare the two mapping techniques.

4.2 Results

The results of the runs were fairly similar. As follows, each result is discussed and the produced map is displayed for comparison purposes. For videos of each of these runs, please see `<https://www.youtube.com/channel/UCBEzz1zxm7b2ZLxDUpxfCcA/feed>` The compared RFID on and off runs are shown side by side for easier comprehension.

4.2.1 Manual Run

Figure 4.3 and Figure 4.4 are maps made while A.T.I.C.U.S.S. was under remote control. Both runs took 20 minutes. These two also best exemplify the disparity between having the RFID reader on and having it off.

4.2.2 Random Run

The maps from the Random runs can be seen in Figure 4.3, showing the run with the RFID Mapper turned off, and in Figure 4.4, where the RFID Mapper is turned on. The run with

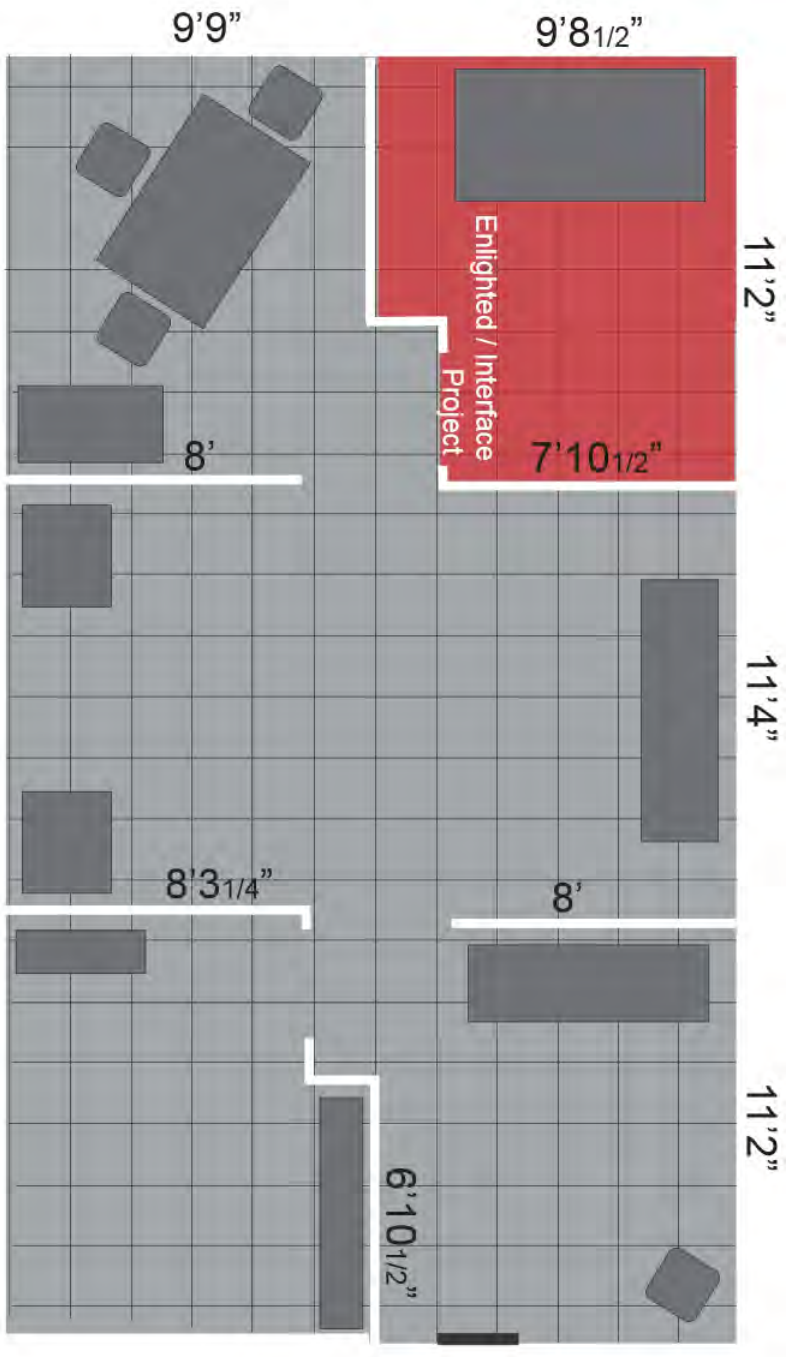

Figure 4.2: Map of the A-smart-ment

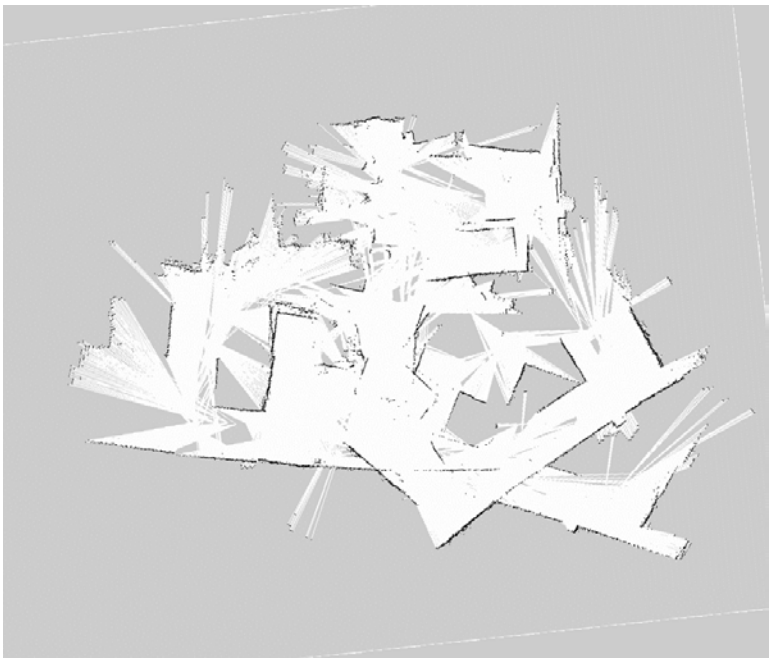

Figure 4.3: Remote Control Run with RFID Mapper turned off

Figure 4.4: Remote Control Run with RFID Mapper turned on

Figure 4.5: Random Run with RFID Mapper turned off

the RFID Mapper turned off ran for 43 minutes and the run with the RFID Mapper on took 24 minutes.

4.2.3 Greedy Run

Figure 4.7 shows the mapping results of the Greedy Algorithm without the RFID Mapper turned on. It took 38 minutes until it was shut down once the map had degraded. Figure 4.8 shows the same algorithm but the the RFID mapper turned on. The result is after 90 minutes of run-time.

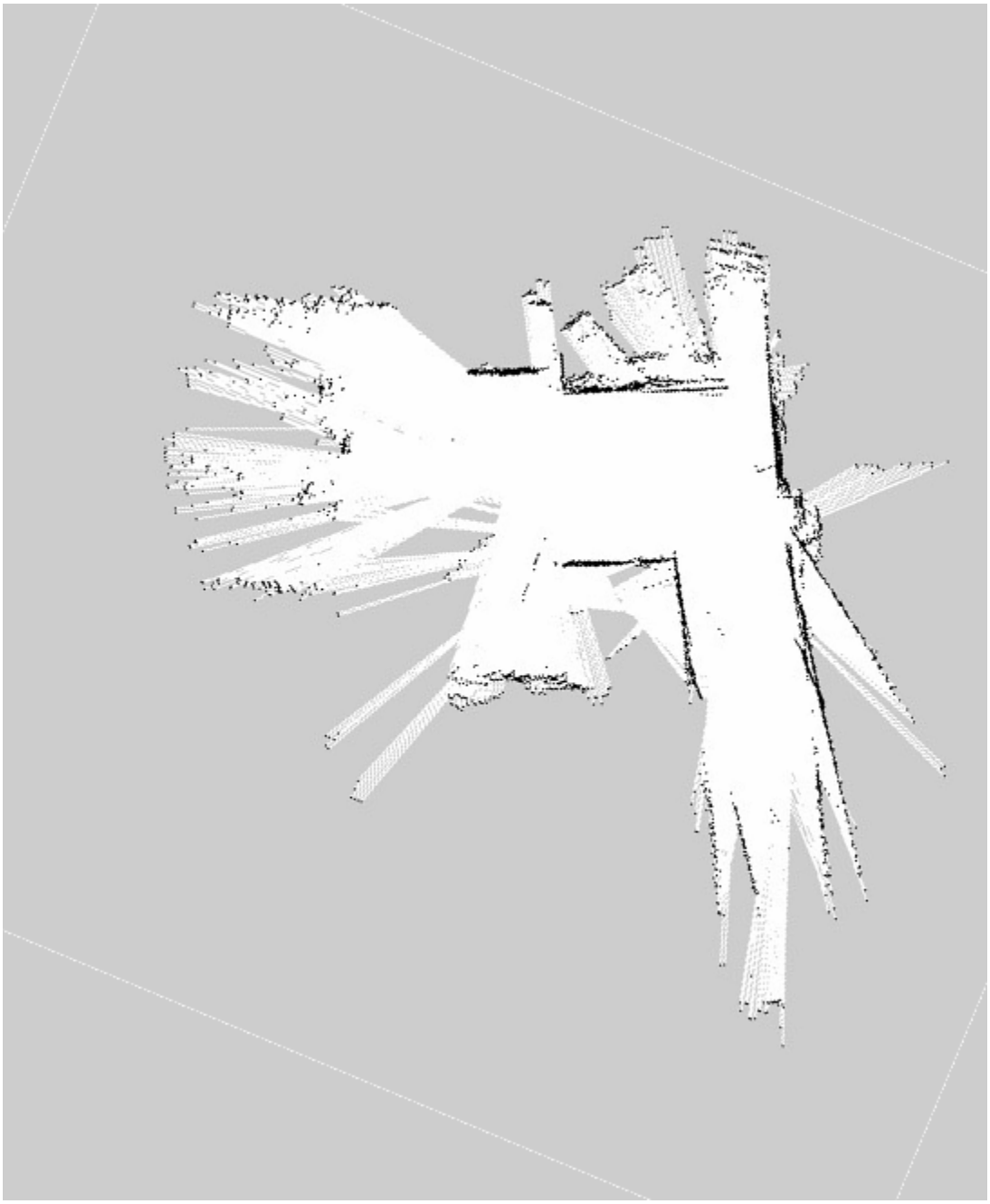

Figure 4.6: Random Run with RFID Mapper turned on

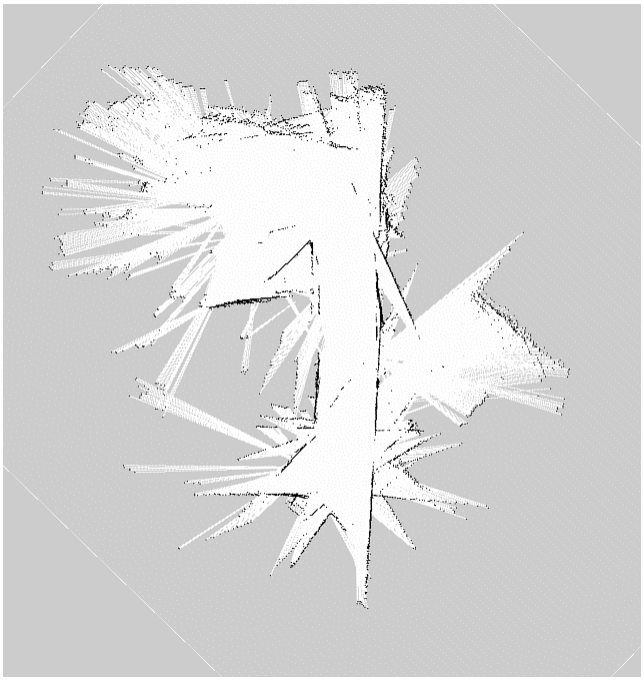

Figure 4.7: Greedy run with RFID Mapper turned off

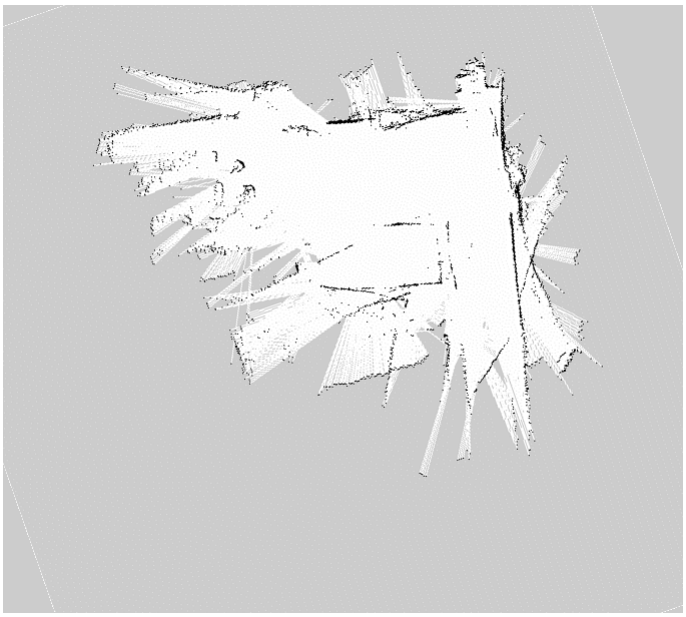

Figure 4.8: Greedy Run with RFID Mapper turned on

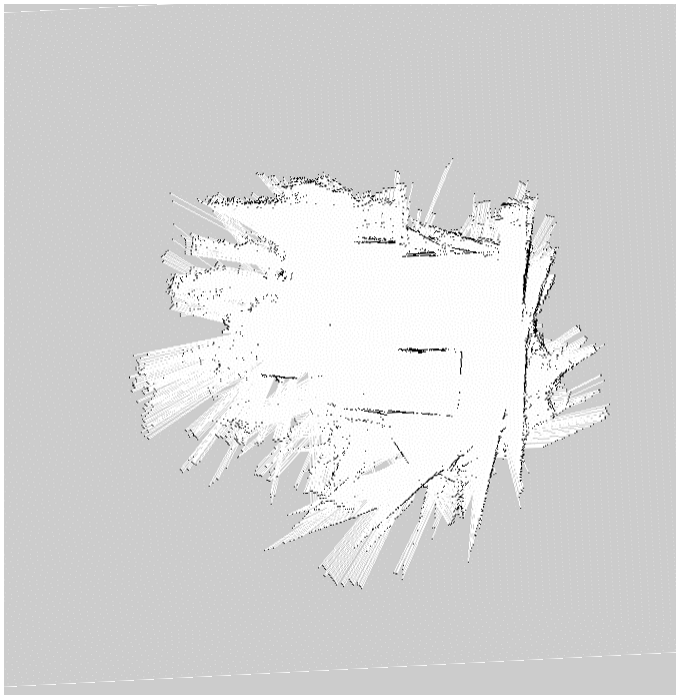

Figure 4.9: Evolved Run with RFID Mapper turned off

4.2.4 Evolved Run

Figure 4.9 and Figure 4.10 show the results for the Evolved Mapping behavior. Without the RFID Mapper turned on, it took 44 minutes. With it on, it lasted 70 minutes.

4.2.5 A-smart-ment

The mapping of the the A-smart-ment was to serve as a quantitative comparison between using the RFID tags for localization and not using them. Therefore, the Hypothesis is that utilizing the RFID tags as landmarks will improve localization efforts. The Null Hypothesis is then that there will be no difference between using the RFID tags or not. The differences in the maps is nearly indistinguishable, it being a small environment. Two maps can be seen in Figures 4.11 and 4.12.

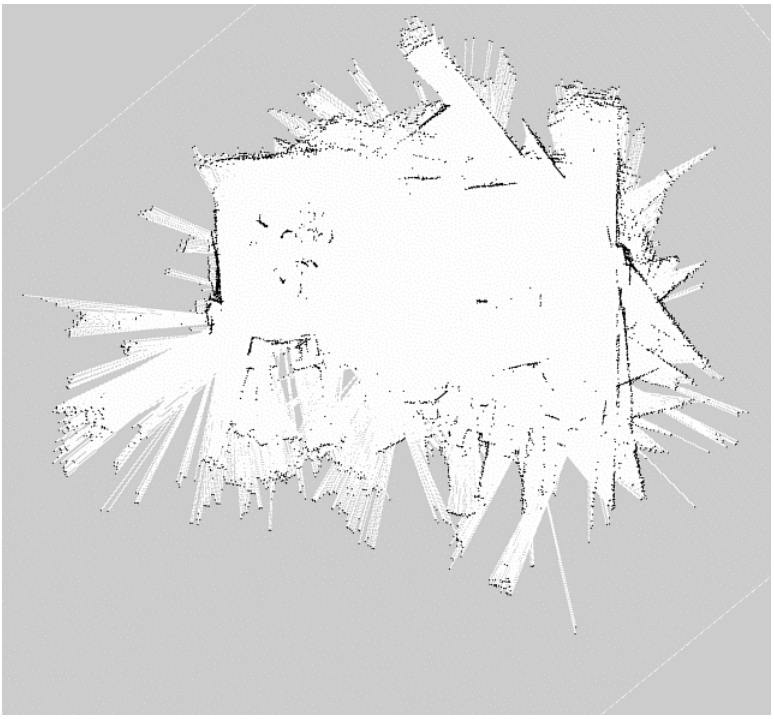

Figure 4.10: Evolved Run with RFID Mapper turned on

The Confidence was collected over time. The averages for the 10 runs with RFID and 10 runs without the RFID can be seen in Figure 4.13.

This graph compares directly the use over 10 runs of each method: using the RFID tags and ignoring them.

The other value mapped during the runs was the Disparity between where the robot thought it was at the time and where the current RFID was telling it was. These values for the 10 runs can be seen in Figure 4.14. The average line for these values can be seen in Figure 4.15.

This shows a trend of increasing Disparity between the robot's localization and the RFID tag coordinates. These results will be discussed in the next section.

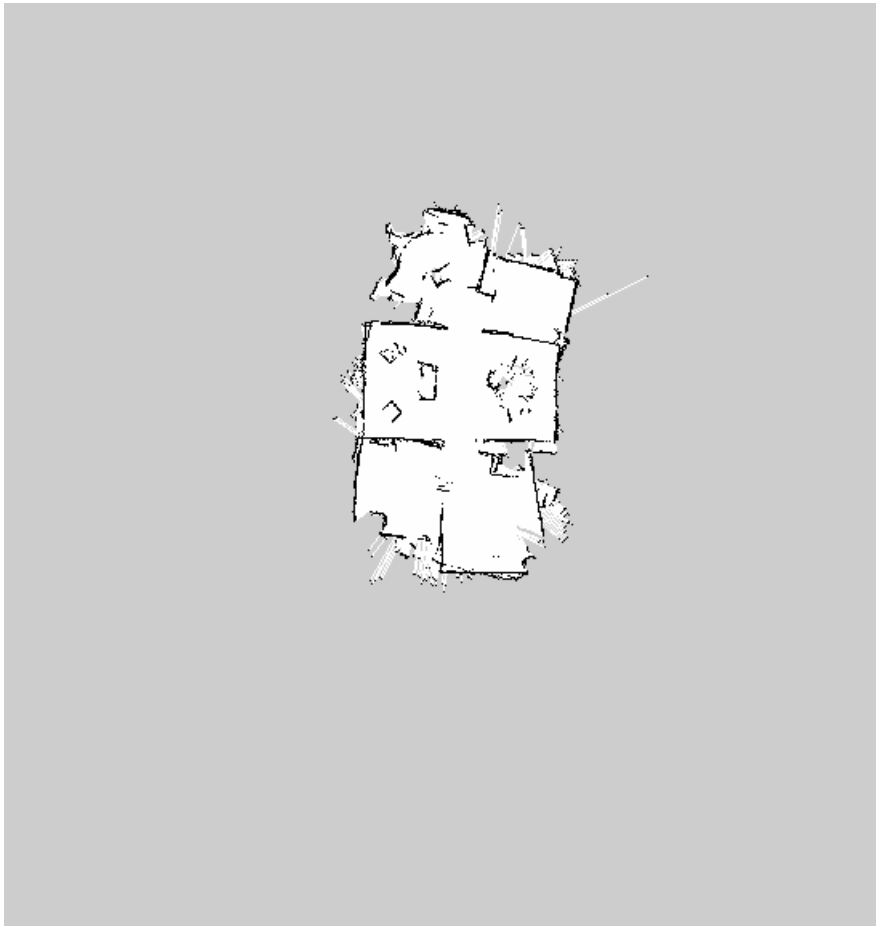

Figure 4.11: Map of A-smart-ment made using RFID tags

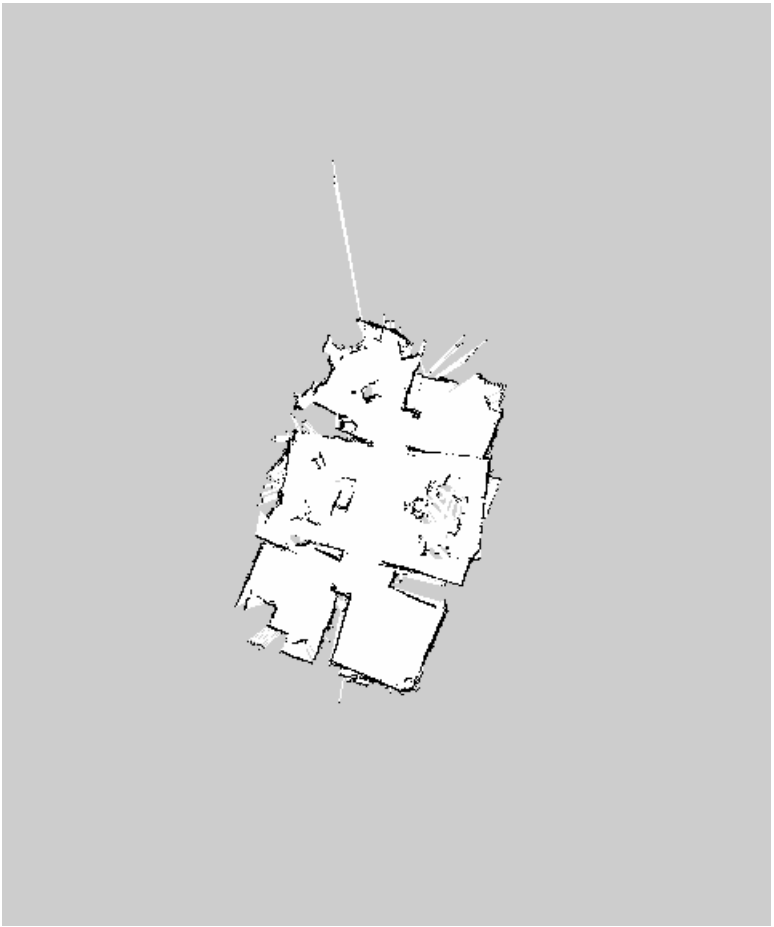

Figure 4.12: Map of A-smart-ment made without using RFID tags

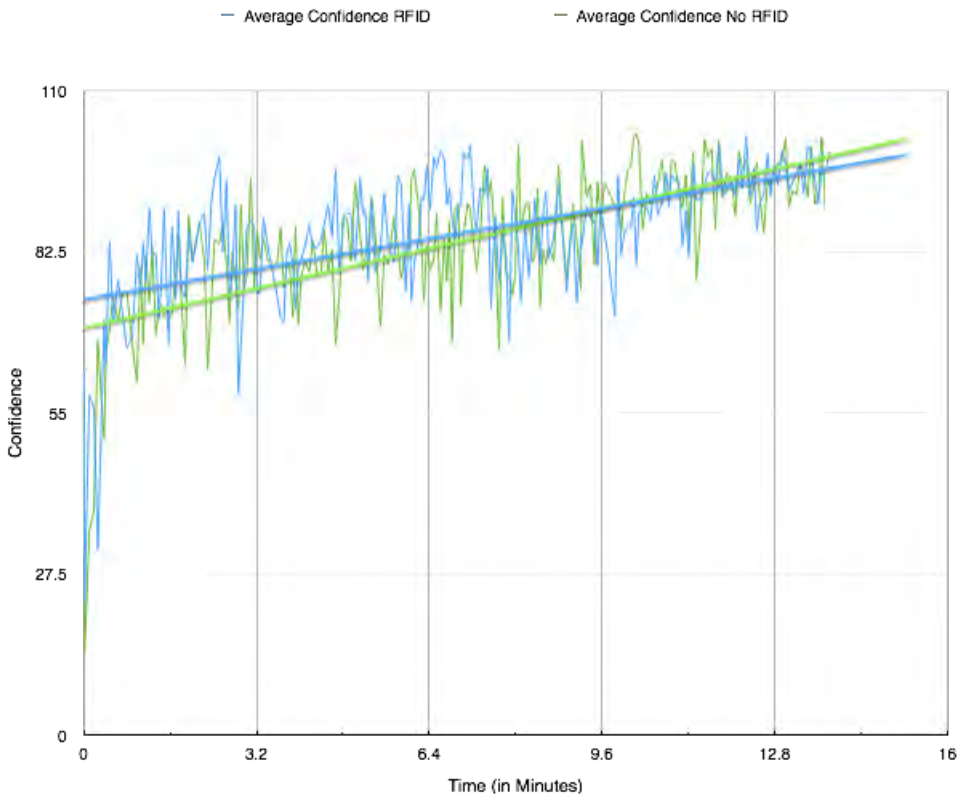

Figure 4.13: Average Confidence over Time, Green being without RFID tags, Blue with RFID

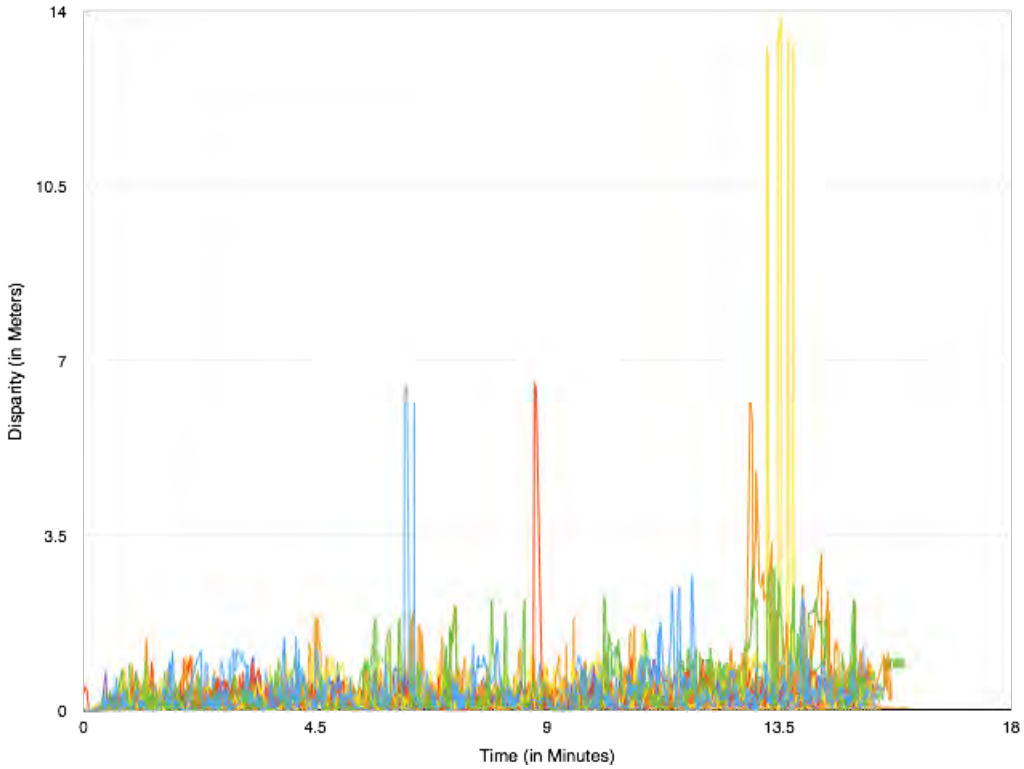

Figure 4.14: Disparity Levels for All 10 Runs using the RFID Tags

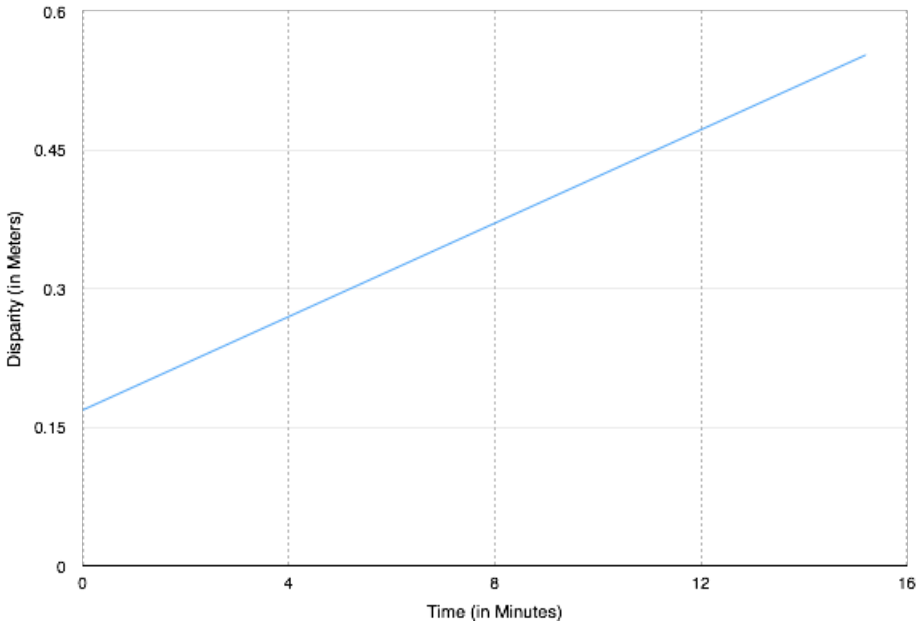

Figure 4.15: Average Disparity over Time

Chapter 5

Discussion, Limitations, and Future Work

5.1 Discussion of Results

As can be seen by the maps, the Turtlebot is not perfect at mapping. In fact, ROS recommended to run a calibration of the odometers and gyro before every mapping run. However, we found this to normally do more harm than good. What can also be seen from the maps is a marked improvement in the quality of the map with the RFID Mapper turned on.

During observations, eventually, all maps deteriorated as the inaccurate odometry misdiagnosed the robot's orientation. The Gmapping Program uses multiple inputs, one of them being a laser scanner. However, the Turtlebot uses a Kinect sensor in lieu of this, primarily because an adequate laser scanner costs several thousands of dollars. The Kinect is only accurate at about 10 meters, and completely degrades after around 20. It is also very noisy, and does not provide enough clarity for accurate scan-matching for the Gmapping algorithm. The RFID tags only give information about location, not orientation. Therefore, the Turtlebot uses odometry information, which is often faulty. As time goes on, inaccurate orientation readings accrue and destroy the map. After this, scan-matching is nearly impossible. This means that eventually, as the same space is driven over and over again, the map accuracy is eventually destroyed.

However, in several cases, the RFID tags allowed the map to reorient itself properly. This can be seen in Figure 5.1. The protrusion of the wall is actually the starting orientation of the map. Areas 1,2,3 on the left are the same area. The same goes for Areas 1 and 2 on the right. This occurs when the detail from the Kinect is not confident enough, so the location is taken from the odometry/gyrometer. On the right, however, the robot was eventually able to

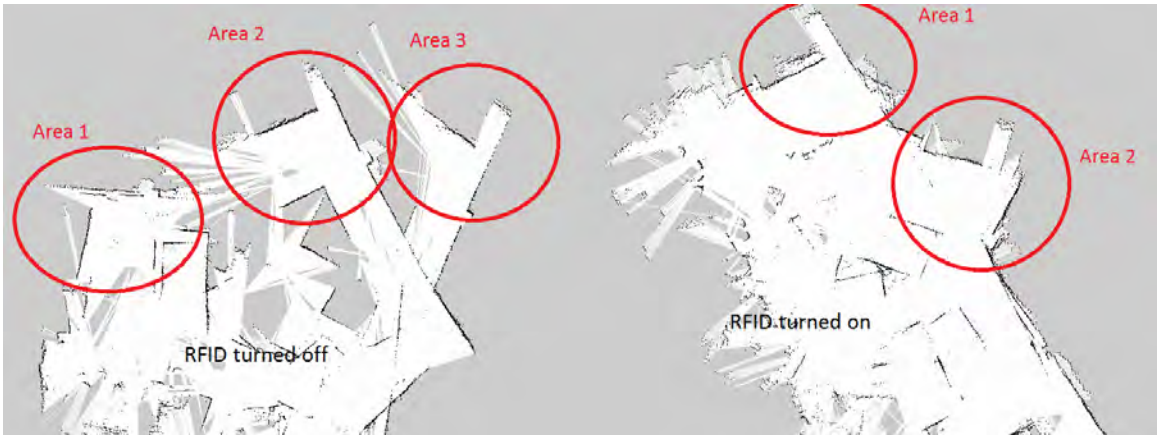

Figure 5.1: Manual Runs Side-by-Side, showing the areas of Repetition

re-orient itself. On the left, it did not, and continued to re-map the same area in a staggered rotation pattern, due to the drift in the odometry/gyrometer.

5.1.1 Comparison of Runs

In order to compare the different maps, each was imported into The GNU Image Manipulation Program, or GIMP. GIMP is simply an open-source photo-editing program. Once here, the differences between the common areas mapped could be seen. As the methods often moved in different directions, the area mapped by all was relatively small. This common area is shown in Figure 5.3.

Once in GIMP and separated into different layers, this common area could be set into Difference mode, where difference between it and another layer show as white and commonalities show as black. An example can be seen in Figure 5.4.

These images could then be compared with a Color Histogram, showing the count for pixels of a certain color in the image. In this case, we are looking for black pixels, representing commonality. Black is 0 on the scale of color saturation, where white is 255. Allowing some room for grey common areas, pixels were counted if they were in the range of 0 to 25. The

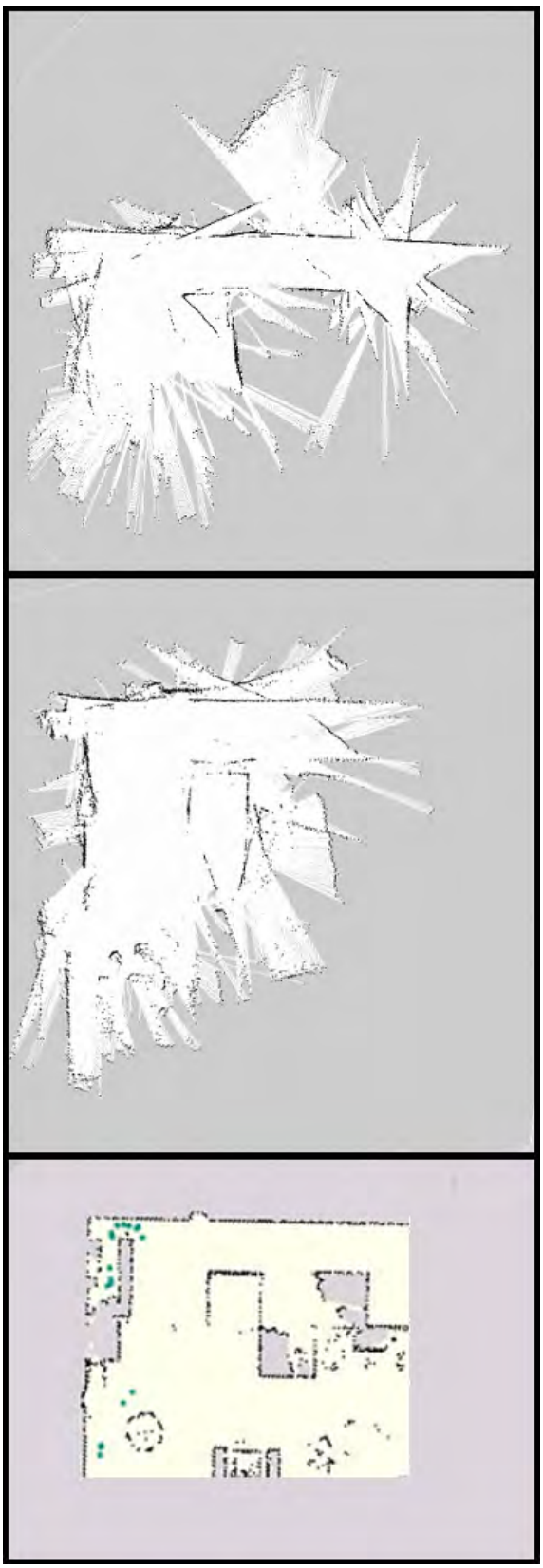

Figure 5.2: Comparison of different methods. From top to bottom: Greedy with RFID turned off, Greedy with RFID turned on, and the SRI Developed Map, zoomed in.

Figure 5.3: Area Mapped by all runs of A.T.I.C.U.S.S.

Figure 5.4: Overlay of Common Area with Greedy without RFID Map

Histogram shows the percentage of pixels in the image fitting 0 to 25. These numbers can be seen here:

Table 5.1: Aware House Map Agreement Percentages

	Without RFID	With RFID
Manual	51.00%	70.20%
Random	67.90%	61.50%
Greedy	68.00%	66.70%
GA Evolved	72.70%	71.70%
Average	64.90%	67.53%

These average percentages show that most of the time, the RFID tags improve map quality, but only slightly. 3% is not a large improvement. In fact, most of the maps compared show a higher correlation when the RFID is turned off. By looking at the maps and seeing an improvement to the average correlation between the SRI map the the RFID maps, we can say that the RFIDs do improve the Maps, but only slightly.

In order to further quantify the differences between using the RFID tags and not, we can look at the differences in Confidences taken from mapping the A-smart-ment. This was done by using a Wilcoxon Signed-Rank Test [43].

A Wilcoxon Signed-Rank Test is useful for determining if the difference between paired results is significant enough to reject the null hypothesis. It is best used for a non-normal distribution and when a random subset of the data is taken, smaller than 20. For the N pairs, the difference is taken between each. Any difference equaling zero is stored, but removed from the N count. The absolute value of these numbers is found, as well as the sign of the difference. Once this is done, these values are ranked according to their absolute value. If there are any ties, they share the average of the ranks they span. The ranks are then multiplied by the signs of the differences. All negatives are added and all positives are added. W is the smaller of the two numbers. If W is less than the critical value for that N, acquired from a Wilcoxon

Signed-Rank Test Critical Values chart, then the result is significant and the null hypothesis can be rejected. If W is larger than that, the null hypothesis fails rejection.

In our case, the paired values are going be the confidence level while using the RFID mapper and the confidence without using the RFIDs. Our null hypothesis is that there is no difference between using the RFID Tags for localization and using them. The 10 runs of each method are paired off to their counter. A random time T is chosen for each pair and the confidence levels at that time are taken. This means N = 10. With N being the sample size, the significant value for W must be less than 8, according to the Wilcoxon Signed-Rank Test. If W is less than 8, then the value is significant. The values for the Wilcoxon Signed-Rank Test can be seen in this table:

Table 5.2: Wilcoxon Signed-Rank Test on Confidence Levels

N	No RFID	RFID	No RFID - RFID	SGN	ABSOLUTE	Rank	Rank * SGN
1	102.2119	99.8958	2.3161	1	2.3161	3	3
2	110.9270	79.6238	31.3031	1	31.3031	9	9
3	112.6569	84.2009	28.4560	1	28.4560	7	7
4	107.3379	113.2259	-5.8880	-1	5.8880	4	-4
5	111.5149	71.5043	40.0106	1	40.0106	10	10
6	101.7490	95.2353	6.5136	1	6.5136	5	5
7	102.4530	104.3779	-1.9249	-1	1.9249	2	-2
8	104.3349	103.0830	1.2519	1	1.2519	1	1
9	112.1880	80.9244	31.2635	1	31.2635	8	8
10	82.9792	110.0240	-27.0447	-1	27.0447	6	-6
							Total
							31
						-Sum	12
						+Sum	43
						W	12

The result of the Wilcoxon Signed-Rank Test is then a W of 12. 12 is greater than 8, therefore, we cannot reject the null hypothesis. This means that according to our results in the A-smart-ment, there is no significant difference between mapping with the RFID tags and mapping without them.

We can also see this in the graphs from our results from the A-smart-ment. While the average confidence while using the RFID tags starts higher than without the tags, eventually not using the tags catches up and surpasses the confidence levels of using the tags. The disparity graph also shows this. Originally, while the map is being built, the RFID Disparity is low. This value increases over time as the known-map becomes more accurate and the map shifts, readjusting the X,Y coordinates of the map, but not the RFID tags. This could also explain why the confidence levels while using the RFID tags are eventually surpassed.

While it's true that we can see no difference in localization using the RFID tags, they do improve the quality of the maps, slightly, as well as improve early localization in the mapping process.

This doesn't mean that RFID tags can be used as location landmarks on their own; it just means that adding the capability to a mapping robot to read RFID tags that it itself mapped does not improve its localization. This also makes sense, from a practical stand-point. When the robot maps a tag, it makes an estimate of its position based on the environment it sees and assigns the tag that location. Later, when re-encountering that tag, it already has an estimated position based on the environment, an environment it has more fully explored than the last time it encountered the tag. This tag then merely represents its past, and SLAM techniques generally mean that this past estimate is going to be more inaccurate than its current estimate as the known map has only improved. The RFID contains no additional information that the robot doesn't already have upon re-encountering it.

Of our original goals:

- Add the capability to read and map RFIDs to the Turtlebot.
- Add autonomous mapping software to the current ROS methods.
- Compare different mapping algorithms.
- Successfully autonomously map an environment using A.T.I.C.U.S.S.
- Provide a map as accurate as given by the SRI robot.

- Show that RFID Tags Improve Localization

all but the last two were achieved. Due to the inaccuracies of the hardware, we do not see accurate maps in a large environment feasible with the platform. The software should allow the robot to build continuously, ever expanding the known map.

5.2 Limitations of A.T.I.C.U.S.S.

The greatest limitation is the battery life; at only two hours, almost no space of meaningful size can be mapped in one run. Even the manual run took nearly a quarter of the robot's battery life, and only mapped a small fraction of the room. This makes our initial goal of a One-Run-Mapper moot.

Another obvious limitation with the methods tested is odometry. Generally, it has a 30% error in measurements. The Turtlebot relies on the odometry of the Create, which is a ten year old design. Not only is it ten years old, but it was always intended to be mass-marketable. Developed by iRobot under Rodney Brooks‘ ideas of reactive robotics, it put Roombas into the homes of thousands of people. However, being completely reactive, it does not need to know exactly where it is. This is not the case for mapping, so becomes a problem when trying to use the Create for such a purpose.

5.3 Future Work

An improvement to the RFID Localization software is needed. The average confidence level while using the RFID tags should not have dropped below that of not using the RFID tags. This seems to indicate they almost detract, once improperly mapped.

In order to fix this, a more adaptive RFID Localizer needs to be implemented. Instead of storing one X,Y value per tag, perhaps multiple values can be stored, like particles. When a tag is read then, it takes an average of all particles for the RFID. This could improve the confidence levels as well as make for a more adaptive and robust RFID Localizer. Pursuit

of this would also improve localization for devices that are using only the RFID tags for localization information, which is still viable.

A known limitation from the beginning was the decision to only use the ID of the RFID tag. With a tag every 50cm in every direction, there are many more possibilities that simply reading the tags ID. While the memory on a tag is small, it could be used to store information, such as where it is, or what room it is in, or directions of nearby landmarks [66]. These ideas were not delved into in the process of developing the robot.

Obviously, A.T.I.C.U.S.S. needs to be improved in order to be able to provide usable maps. The first step would be better odometry. Luckily, or unluckily, in terms of this thesis, Willow Garage has developed a new Turtlebot, due to release sometime before the end of the year. It has an integrated gyro, which does not require calibration every run, improved odometry, and integrated power-out ports to power and charge a laptop. This also means that the Turtlebot can now charge itself and the laptop controlling it at the same time, in a manner similar to a Roomba. It also has additional battery pack ports, to extend its run time. But, it can still charge itself, which is a vast improvement. These factors alone could improve A.T.I.C.U.S.S. enough to produce usable maps.

A.T.I.C.U.S.S. could also be made more ubiquitous with these changes. If it can charge itself, once it finishes mapping an office, it can be used as an office assistant or tour guide, of sorts. Voice recognition is a package of ROS, and while we did not use it for this thesis, we did install it and get it running and used it to control A.T.I.C.U.S.S. with simple voice commands. With RFID integrated into the maps, tags could be assigned callable locations, such as “Bob‘s Office.” Then, when you have the robot take a stack of papers to Bob, it simply navigates to a location of an aliased RFID tag.

Another obvious next step is to have multiple robots. Since this is intended for an office-like environment, a kind of all-purpose robot, that provides security, information, and maintenance monitoring, multiple robots would be employed anyway. This would allow more area to be explored quickly.

Chapter 6

Conclusion

SLAM is a major issue for current robotics. Through the combination of better sensors and refined algorithms, robots can know where they are in relation to their environment. This is necessary for any type of multi-location goal.

By combining Genetic Programming with Behavior Based Robotics, robots can be evolved in order to have better exploratory behaviors, allowing them to build better maps through SLAM.

Through the development of the SCOUT flooring system, Interface saw an opportunity to aid in SLAM map building, by adding yet another sensor; fixed location RFIDs. These provide landmarks that can re-orient the robot. A.T.I.C.U.S.S. uses the combination of many techniques in order to map his environment. Genetic Programs evolve different behaviors to each type of situation encountered in the Aware House; halls, corners, and open spaces. These differences are determined by applying a Randomized Hough Transform to the current map.

A.T.I.C.U.S.S. successfully evolved RFID localization abilities and exploration behaviors. However, due to drift of the odometry and gyrometer, the mapping results were limited. RFIDs do add another measurement for accurate localization, however, without accurate orientation information, these benefits are minor, yet still provide enough of a boost to the initial phases of mapping.

A.T.I.C.U.S.S. was able to successfully evolve a decision process on when to use fixed location RFIDs and when to remap their stored location. This same technique can be done to other types of landmarks, whether they are images or structures. This is very important

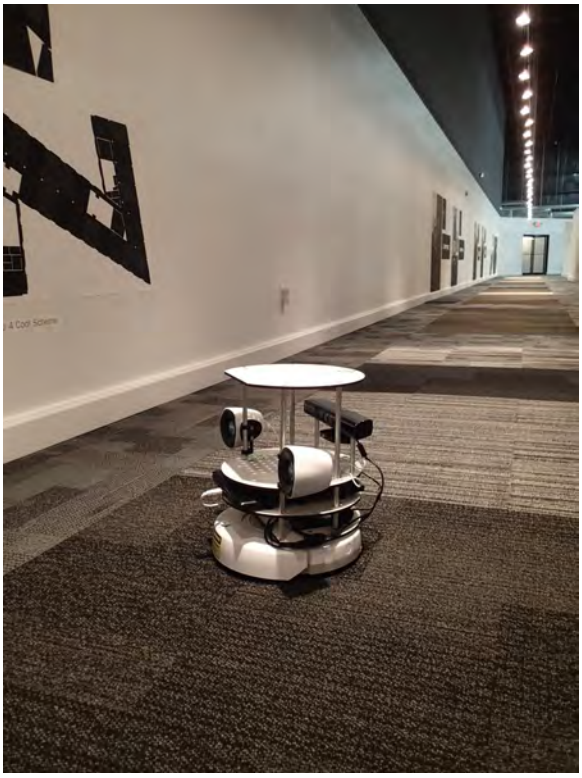

Figure 6.1: A.T.I.C.U.S.S. Exploring the Aware House

in areas where GPS intermittent or not available at all. By adjusting the shape of the map based on landmarks, better maps can be made.

Bibliography

[1] Albiez, J.; Luksch, T.; Berns, K.; and DIllmann, R. (2003) *A Behaviour Network Concept for Controlling Walking Machines.* Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines, Kyoto, March 4-8, 2003. pp. 237-246.

[2] Arkin, R.C. (1998) *Behavior-Based Robotics.* The MIT Press, Cambridge, Massachusetts. 1998.

[3] Bailey, T. and Durrant-Whyte, H. (2006) *Simultaneous Localisation and Mapping (SLAM): Part II.* IEEE Robotics and Automation Magazine. Vol. 13, Iss. 3. pp. 108-117.

[4] Beyer, H.-G. and Schwefel, H.-P.(2002) *Evolution strategies.* Natural Computing Vol. 1, No. 1. pp. 3-52.

[5] Brooks, R. (1992) *Artificial Life and Real Robots.* "Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life," Francisco J. Varela and Paul Bourgine, eds., MIT Press, Cambridge, MA, 1992, pp. 3-10.

[6] Burrattini, E. and Rossi, S. (2010) *Periodic Activations of Behaviours and Emotional Adaptation in Behaviour-based Robotics.* Connection Science. Vol. 22, No. 3. pp. 197-213.

[7] Correll, N.; Arechiga, N.; Bolger, A.; Bollini, M.; Charrow, B.; Clayton, A.; Dominguez, F.; Donahue, K.; Dyar, S.; Johnson, L.; Liu, H.; Patrikalakis, A.; Robertson, T.; Smith, J.; Soltero, D.; Tanner, M.; White, L.; and Rus, D. (2009) *Building a Distributed Robot Garden.* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009. pp. 1509-1516.

[8] Dellaert, F.; Fox, D.; Burgard, W.; and Thrun, S. (1999) *Monte Carlo Localization for Mobile Robots.* 1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings. Vol. 2. pp. 1322-1328.

[9] De Rainville, F.-M.; Fortin, F.-A.; Gardner, M.-A.; Parizeau, M.; and Gagné, C. (2012) *DEAP: A Python Framework for Evolutionary Algorithms.* GECCO Companion '12 Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion. pp. 85-92.

[10] Deyle, T.; Nguyen, H.; Reynolds, M.S.; and Kemp, C.C. (2010) *RFID-Guided Robots for Pervasive Automation.* Pervasive Computing. Vol. 9, No. 2. pp. 37-45.

[11] Dorigo, M. (1993) *Genetics-Based Machine Learning and Behavior-Based Robotics: A New Synthesis.* IEEE Transactions on Systems, Man and Cybernetics. Vol. 23, Iss. 1. pp. 141-154.

[12] Egerstedt, M. (2000) *Behavior Based Robotics Using Hybrid Automata.* Hybrid Systems: Computation and Control Lecture Notes in Computer Science, 2000. Vol. 1790/2000. pp. 103-116.

[13] Fallon, M. F.; Johannsson, H.; and Leonard, J. J. (2012). *Efficient scene simulation for robust Monte Carlo localization using an RGB-D camera.* Robotics and Automation (ICRA), 2012 IEEE International Conference on. pp. 1663-1670.

[14] Feder, H.J.S.; Leonard, J.J.; and Smith, C.M. (1999) *Adaptive Mobile Robot Navigation and Mapping.* The International Journal of Robotics Research July 1999. Vol. 18, No. 7. pp. 650-668.

[15] Fietz, A.; Jakisch, S.M.; Visel, B.A.; and Fritsch, D.(2010) *Automated 2D Measuring of Interiors Using a Mobile Platform.* Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010). pp. 115-120.

[16] Floreano, D. and Mondada, F.(1996) *Evolution of Homing Navigation in a Real Mobile Robot.* IEEE Transactions on Systems, Man and Cybernetics Part B : Cybernetics. Vol. 26, Num. 3. pp. 396-407.

[17] Floreano, D. and Mondada, F. (1998) *Evolutionary Neurocontrollers for Autonomous Mobile Robots.* Neural Networks. Vol. 11. pp. 1461-1478.

[18] Floreano, D. and Urzelai, J. (2000) *Evolutionary Robots with On-line Self-Organization and Behavioral Fitness.* Neural Networks. Vol. 13, Iss. 4-5. pp. 431-443.

[19] Floreano, D. and Keller, L. (2010) *Evolution of Adaptive Behaviour in Robots by Means of Darwinian Selection.* PLoS Biology. Vol. 8, Iss. 1. e1000292.

[20] Frese, U. (2006) *Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping.* Autonomous Robots. Vol. 21, Number 2. pp.103-122.

[21] Fox, D., Burgard, W., and Thrun, S. (1997) *The Dynamic Window Approach to Collision Avoidance.* IEEE Robotics and Automation. Vol. 4, Iss. 1. pp. 23-33.

[22] Ganganath, N. and Leung, H. (2012) *Mobile robot localization using odometry and kinect sensor.* 2012 IEEE International Conference on Emerging Signal Processing Applications (ESPA). pp. 91-94.

[23] Grisetti, G., Stachniss, C., and Burgard, W. (2007) *Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters.* IEEE Transactions on Robotics. Vol. 23, Iss. 1. pp. 34-46.

[24] Guivant, J.E. and Nebot, E.M. (2001) *Optimization of the Simultaneous Localization and Map-building Algorithm for Real-time Implementation.* IEEE Transactions on Robotics and Automation. Vol. 17, Iss. 3. pp. 242-257.

[25] Hahnel, D.; Burgard, W.; Fox, D.; Fishkin, K.; and Philipose, M.(2004) *Mapping and Localization with RFID Technology.* ICRA '04. 2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. Vol 1. pp. 1015-1020.

[26] Harvey, I.; Husbands, P.; and Cliff, D. (1993) *Issues in Evolutionary Robotics.* Proceedings of the Second International Conference on From Animals to Animats 2 : Simulation of Adaptive Behavior: Simulation of Adaptive Behavior. pp. 364-373.

[27] Horswill, I.D. (2000) *Functional Programming of Behavior-Based Systems.* Autonomous Robots. Vol. 9, Iss. 1, August 2000. pp. 83-93.

[28] Huq, R.; Mann, G.K.I.; and Gosine, R.G. (2006) *Behavior-Modulation Technique in Mobile Robotics Using Fuzzy Discrete Event System.* IEEE Transactions on Robotics. Vol. 22, No. 5, October 2006. pp. 903-916.

[29] Husbands, P.; Harvey, I.; Cliff, D.; and Miller, G.(1997) *Artificial Evolution: A New Path for Artificial Intelligence?* Brain and Cognition 34. pp. 130-159.

[30] Illingworth, J. and Kittler, J. (1988) *A Survey of the Hough Transform.* Computer Vision, Graphics, and Image Processing. Vol. 44, Iss. 1. pp. 87-116.

[31] Joho, D.; Plagemaan, C.; and Burgard, W. (2009) *Modeling RFID Signal Strength and Tag Detection for Localization and Mapping.* ICRA '09. IEEE International Conference on Robotics and Automation, 2009. pp. 3160-3165

[32] Karafotias, G.; Haasdijk, E.; and Eiben, A.E. (2011) *An Algorithm for Distributed Online, On-board Evolutionary Robotics.* GECCO '11 Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. pp. 171-178.

[33] Keymeulen, D.; Iwata, M.; Kuniyoshi, Y.; and Higuschi, T. (1998) *Online Evolution for a Self-Adapting Robotic Navigation System Using Evolvable Hardware.* Artificial Life. Vol. 4, No. 4. pp. 359-393.

[34] Kiryati, N.; Eldar, Y.; and Bruckstein, A.M. (1991) *A probabilistic Hough transform.* Pattern Recognition. Vol. 24, Iss. 4. pp. 303-316.

[35] Kleiner, A.; Prediger, J.; and Nebel, B.(2006) *RFID Technology-based Exploration and SLAM for Search And Rescue.* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006. pp. 4054-4059.

[36] Koulouriotis, D.E.; Diakoulakis, I.E; and Emeris, D.M. (2001) *Learning Fuzzy Cognitive Maps using Evolution Strategies: a Novel Schema for Modeling and Simulating High-Level Behavior.* Proceedings of the 2001 Congress on Evolutionary Computation. Vol. 1. pp 364-371.

[37] Kulyukin, V.; Gharpure, C.; Nicholson, J.; and Pavithran, S.(2004) *RFID in Robot-Assisted Indoor Navigation for the Visually Impaired.* 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. Vol. 2. pp. 1979-1984.

[38] Lee, M. (2003) *Evolution of behaviors in autonomous robot using artificial neural network and genetic algorithm.* Information Sciences 155. pp. 43-60.

[39] Lin, F. and Ying, H. (2002) *Modeling and control of fuzzy discrete event systems.* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics. Vol. 32, Iss. 4. pp. 408-415.

[40] Marder-Eppstein, E.; Berger, E.; Foote, T.; Gerkey, B.; and Konolige, K. (2010) *The Office Marathon: Robust navigation in an indoor office environment.* 2010 IEEE International Conference on Robotics and Automation (ICRA). pp. 300-307.

[41] Mataric, M.J.(1992) *Integration of Representation Into Goal-Driven Behavior-Based Robots.* IEEE Transactions on Robotics and Automation. Vol. 8, Iss. 3. pp. 304-312.

[42] Matas, J.; Galabos, C.; and Kittler, J. (1998) *Progressive Probabilistic Hough Transform.* In M. S. Nixon, editor, Proc British Machine Vision Conference. pp. 256-265.

[43] Nehmzow, U. (2006)*Scientific Methods in Mobile Robotics: Quantitative Analysis of Agent Behaviour.* Springer-Verlag, London, England.

[44] Nicolescu, M.N. and Matarić, M.J. (2002) *A hierarchical Architecture for Behavior-Based Robots.* AAMAS '02 Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1. pp. 227-233.

[45] Nordin, P. and Banzhaf, W. (1997) *An On-Line Method to Evolve Behavior and to Control a Miniature Robot in Real Time with Genetic Programming.* Adaptive Behavior, January 1997. Vol. 5, No. 2. pp. 107-140.

[46] Pfaff, P.; Burgard, W.; and Fox, D. (2006) *Robust Monte-Carlo Localization using Adaptive Likelihood Models.* European Robotics Symposium 2006. Springer Tracts in Advanced Robotics, 2006. Vol. 22/2006. pp. 181-194.

[47] Proetzsch, M.; Luksch, T.; and Berns, K. (2005) *Fault-Tolerant Behavior-Based Motion Control for Offroad Navigation.* 20th IEEE International Conference on Robotics and Automation. pp 18 - 22.

[48] Proetzsch, M.; Luksch, T.; and Berns, K. (2010) *Development of complex robotic systems using the behavior-based control architecture iB2C.* Robotics and Autonomous Systems 58 (2010). pp. 4667.

[49] Qiu, D. (2005) *Supervisory Control of Fuzzy Discrete Event Systems: A Formal Approach.* IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics. Vol. 35, Iss. 1. pp. 72-88.

[50] Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. (2009) *ROS: an open-source Robot Operating System.* International Conference on Robotics and Automation,ser. Open-Source Software Workshop, 2009.

[51] Radovnikovich, M,; Lorenz, L.; Hellenbeck, K.; Grzebyk, S.; Truitt, M.; Norman, M.; Abdo, P.; Vasquez, O.; Iyengar, K.; Bowman, R.; Parker, R.; and Suriano, J.(2012) *Okland University Proudly PResents: Botzilla.* IGVC 2012.

[52] Rückert, E. (2009) *Simultaneous Localization and Mapping for Mobile Robots with Recent Sensor Technologies.* Beschluss der Curricula-Kommission fr Bachelor-, Master- und Diplomstudien vom 10.11.2008.

[53] Russell, S. and Norvig, P. (2003) *Artificial Intelligence A Modern Approach. 2nd Ed.* Pearson Education, Inc., Upper Saddle River, New Jersey.

[54] Schiele, B. and Crowley, J.L. (1994) *A comparison of position estimation techniques using occupancy grids.* Robotics and Autonomous Systems. Vol. 12, Iss. 34, April 1994. pp.163-171.

[55] Schmidt, D.; Luksch, T.; Wettach, J.; and Berns, K.(2006) *Autonomous Behavior-Based Exploration of Office Environments.* In 3rd International Conference on Informatics in Control, Automation and Robotics Icinco. pp. 235-240.

[56] Sim, R. and Roy, N. (2005) *Global A-Optimal Robot Exploration in SLAM .* Proceedings of the IEEE/RSJ International Conference of Robotics and Automation (ICRA 2005). pp. 661-666.

[57] Sprong, C. (2011) *Common tasks in Evolutionary Robotics, an overview.*

[58] Surmann, H.; Nüchter, A.; and Hertzberg, J. (2003) *An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments.* Robotics and Autonomous Systems. Vol. 45, Iss. 34. pp. 181-198.

[59] Tardòs, J.D.; Neira, J.; Newman, P.M.; and Leonard, J.J. (2002) *Robust Mapping and Localization in Indoor Environments Using Sonar Data.* The International Journal of Robotics Research April 2002. Vol. 21, No. 4. pp. 311-330.

[60] Thrun, S. and Bücken, A. (1996) *Integrating grid-based and topological maps for mobile robot navigation.* Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96). pp. 944-950.

[61] Thrun, S. (1998) *Learning metric-topological maps for indoor mobile robot navigation.* Artificial Intelligence. Vol. 99, Iss. 1. pp. 21-71.

[62] Thrun, S; Fox, D.; Burgard, W.; and Dellaert, F. (2000) *Robust Monte Carlo Localization for mobile Robots.* Artificial Intelligence. Vol. 128, No. 1-2. pp. 99-141.

[63] Thrun, S.; Burgard, W.; and Fox, D. (2006) *Probabilistic Robotics.* The MIT Press, Cambridge, Massachusetts.

[64] Trujillo, L.; Olague, G.; Lutton, E.; Fernández de Vega, F.; Dozal, L; and Clemente, E.(2011) *Speciation in Behavioral Space for Evolutionary Robotics.* Journal of Intelligent and Robotic Systems. Vol. 64, No. 3-4. pp. 323-351.

[65] Vorst, P.; Schneegans, S.; Yang, B.; and Zell, A. (2008) *Self-Localization with RFID snapshots in densely tagged environments.* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008. pp. 1353-1358.

[66] Willis, S. and Helal, S. (2005) *RFID Information Grid for Blind Navigation and Wayfinding.* Ninth IEEE International Symposium on Wearable Computers, 2005. Proceedings. pp. 34-37.

[67] Xu, L.; Oja, E.; and Kultanen, P.(1990) *A new curve detection method: Randomized Hough transform (RHT).* Pattern Recognition Letters. Vol. 11, Iss. 5. pp. 331-338.

[68] Xu, L. and Oja, E.(1993) *Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities.* Image Understanding. Vol. 57, No. 2. pp. 131-154

[69] Xu, L. and Oja, E.(2008) *Randomized Hough Transform.* Encyclopedia of Artificial Intelligence, Ed. by J. R. R. Dopico, J. Dorado, and A. Pazos. pp. 1354-1361.

[70] Yamauchi, B. (1997) *A frontier-based approach for autonomous exploration.*1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings. pp. 146-151.

[71] Zah, C.H and Fezer, S.F. (2008)*Embedded sensor SCOUT flooring system by Interface Flor.* IEEE International Conference on Technologies for Practical Robot Applications, 2008. TePRA 2008. pp. 106-110.

[72] Zaman, S.; Slany, W.; and Steinbauer, G. (2011) *ROS-based Mapping, Localization and Autonomous Navigation using a Pioneer 3-DX Robot and their Relevant Issues.* 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC). pp. 1-5.