CERIAS Tech Report 2012-07

An Agent-Based Model for Navigation Simulation in a Heterogeneous Environment

by Teresa A. Shanklin
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Teresa A. Shanklin

Entitled AN AGENT-BASED MODEL FOR NAVIGATION SIMULATION IN A HETEROGENEOUS ENVIRONMENT

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Eric T. Matson
Chair

Brandeis Marshall

Alejandra Magana

Dong-Han Kim

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Eric T. Matson

Approved by: James Mohler 4/18/2012
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:
AN AGENT-BASED MODEL FOR NAVIGATION SIMULATION IN A HETEROGENEOUS ENVIRONMENT

For the degree of Doctor of Philosophy

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Teresa A. Shanklin
Printed Name and Signature of Candidate

4/18/2012
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

AN AGENT-BASED MODEL FOR NAVIGATION SIMULATION IN A HETEROGENEOUS ENVIRONMENT

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Teresa A. Shanklin

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2012

Purdue University

West Lafayette, Indiana

Dedicated to my husband, daughter, family and friends. Without all your love and support, I would never have finished this. Thanks to all members of 'Team Shanklin' both operational and spirit crew!

ACKNOWLEDGMENTS

I wish to gratefully acknowledge my thesis committee for their insightful comments and guidance. I wish to acknowledge my chair, Professor Eric Matson, for serving as my principal advisor through this journey, and for all his thoughtful guidance. He has allowed me to venture into new areas and develop new passions for learning.

Additionally, I would like to acknowledge the assistance of Professor Brandeis Marshall. She was instrumental in allowing me to complete the research as well as the statistical interpretation. Her time and input were invaluable, and I cannot fully express my gratitude for her assistance.

A special thank you to Professor Alejandra Magana for pushing me to put forth the best document possible. I am thankful for the feedback on critical thought and analysis and feel the research is better for it.

Also, I would like to acknowledge Professor Dong-Han Kim. Although he works out of South Korea, he has graciously given his time to help me complete this milestone.

I want to thank 'Team Shanklin' for their assistance through this adventure. In particular to my parents, Peter Aschenbrenner and Mitchel Friday, thanks for the many, many hours of proof-reading and brainstorming sessions. Without them this day would not have arrived, and even if it had all the commas would be in the wrong place.

A strong thank you to my many friends who have offered their friendship, advice, and support during this long trek. It was easier to concentrate on the research, knowing my family was being tended to.

Finally, a special thanks to my husband James, and my daughter Molly. They have generously given up many hours, days, and weekends of time to allow me to

achieve this goal. I hope I have made you proud. This has been a long time in the making. Albert Einstein said it best, "The important thing is not to stop questioning. Curiosity has its own reason for existing..".

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ABM	Agent-Based Modeling
AIS	Autonomic Information Systems
AMAS	Adaptive Multi-agent System
AOS	Android Operating System
API	Application Programming Interface
CC	Cartesian Coordinate
CIT	Computer and Information Technology
COT	College of Technology
CSE	Complex System Engineering
DEVS	Discrete Event Simulation
DoD	Department of Defense
DOF	Degree's of Freedom
GPS	Global Positioning System
HCI	Human Computer Interface
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
iOS	iPhone Operating System
JADE	Java Database Engine
JVM	Java Virtual Machine
kNN	K Nearest Neighbor
KNOY	Maurice G. Knoy Hall of Technology
LWSN	Richard & Patricia Lawson Computer Science Building
MaNG	Multi-agent Navigation Graph
MaS	Multi-agent System

MaSD	Multi-agent System Development
MaSE	Multi-agent System Engineering
MaSoS	Multi-agent System-of-Systems
MEMS	Micro Electric Mechanical System
NN	Nearest Neighbor
NNS	Nearest Neighbor Search
OS	Operating System
OSI	Open Systems Interconnect
PDR	Pedestrian Dead Reckoning
PPP	Point-to-Point Protocol
RFID	Radio Frequency Identification
RSS	Received Signal Strength
SDK	System Development Kit
SoS	System-of-Systems
SoSE	System-of-Systems Engineering
TDOA	Time Difference Of Arrival
TOA	Time of Arrival
VM	Virtual Machine
Wifi	Wireless Fidelity

ABSTRACT

Shanklin, Teresa A. Ph.D., Purdue University, May 2012. An Agent-Based Model for Navigation Simulation in a Heterogeneous Environment . Major Professor: Eric T. Matson.

Complex navigation (e.g. indoor and outdoor environments) can be studied as a system-of-systems problem. The model is made up of disparate systems that can aid a user in navigating from one location to another, utilizing whatever sensor system or information is available. By using intelligent navigation sensors and techniques (e.g. RFID, Wifi, GPS, embedded sensors on a mobile device, IMU, etc.) and adaptive techniques to switch between them, brings the possibility of an end-to-end navigational multi-agent system-of-systems (MaSoS).

Indoor location-based applications have a broad appeal for development in navigation, robotics, gaming, asset tracking, networking, and more. GPS technologies have been successfully leveraged for outdoor navigation, but often lose effectiveness indoors due to a more constrained environment, possible loss of signal, lack of elevation information and need for better accuracy.

Increasingly complex problems in navigation allow for the development of a framework for a system-of-systems. Individual systems contain distributed and heterogeneous components that are disparate in nature.

Multiple prototypes and a framework for a multi-agent system-of-systems are presented. The purpose of the model is to overcome the limitations of a single technology navigation system. The system creates a classic system-of-systems utilizing existing and developing localization services. The system provides point-to-point path planning and navigation regardless of the transportation medium, location of the user or current environment.

CHAPTER 1. INTRODUCTION

This study presents an agent-based application, which, as modeled, offers enhanced personal navigation through a dense urban environment. The research develops a framework for an autonomous navigation system. The system-of-systems approach offers individual systems that are unique, distributed and disparate, [1], which overcomes the limitations of a single-technology navigation system. The research sets, as its ultimate goal, creation of a classic system-of-systems by utilizing three existing localization technologies, while offering a common interface to exploit all three services. This interface provides the navigator with end-to-end path-planning and navigation regardless of the transportation medium, location of the user, current connectivity or immediate environment. While both indoor and outdoor navigation have been studied often and recently, to this author's knowledge, using an agent- or multi-agent-based system-of-systems (MaSoS) to design an autonomous navigation system fusing existing signal technologies is unique.

In 1999 Maier presented his seminal work, which proposed five characteristics to identify a System-of-Systems (SoS). [1]

- Operational Independence - Each of the navigational systems operate independently, i.e. the system currently navigating (GPS) is not influenced in its path-planning by knowledge of any other system (Wifi, RFID).
- Managerial Independence - Each of the systems is managed independently.
- Evolutionary Development - The localization information may change over time as routes change.
- Emergent Behavior - The various outputs from localized systems will influence decisions made and routes selected.

- Geographic Distribution - The systems all cover disparate geographic locations and will exchange information as part of the input and decision-making information.

According to DeLaurentis, Crossley and Mane [2], "the first three characteristics primarily describe the problem boundaries and mechanics of the interacting elements while the latter two describe overall behavior." By applying these five definitive characteristics this research is categorized as a classic System-of-Systems; a complex system with multiple, independent systems that interact at various levels.

1.1 Motivation

The majority of research in navigation and localization focuses on individual technologies or a hybrid of several meshed together. These may be broken down into: robotics, assistive devices (blind, cognitive disorders), informational navigation, asset tracking, gaming, etc. Often the research is further subdivided into indoor or outdoor environments; selected mode of transportation; or limitations and goals. The motivation of this research is to construct a multi-agent system-of-systems (MaSoS) to actualize heterogeneous personal navigation.

An example of an extendable SoS, which allows a user to navigate autonomously through available modes of localization, may be seen in the movement of an individual from an indoor RFID localization system, to an outdoor GPS navigation autonomously; or, the SoS might provide location information from one's home to a destination in a distant urban location, taking available modes of transportation.

At present the availability of localization and navigation information are dependent on whatever device is at hand, and is further limited by a user's physical location, network coverage or mode of transportation. The SoS model developed, along with the prototype constructed and tested, combines disparate localization technologies to offer the possibility of an end-to-end solution. The potential benefits of this system include, for example:

1. allowing directional acuity without a priori knowledge;
2. building additional advances for persons with disabilities;
3. building environmental knowledge for alternative routing in traffic tracking and emergency preparedness.

1.2 Statement of Purpose

This work critically examines why an agent-based or multi-agent-based system is appropriate for a navigation application in a heterogeneous environment. Multi-modal movement, through and around multi-story buildings in an urban environment is modeled. The research provides a formalized description of the systems and their model. It introduces an agent-based model within an SoS to allow for the seamless addition of new location and navigation technologies. Simulations and prototypes are developed to aid in the study. This research quantifies measures by looking at which system is active and providing information, and the speed of that system. The research is novel as it looks at both the fusion of existing sensor technologies and a framework to allow the addition of new technologies. These can be seen in the models and prototypes described in Chapter 3.

Instances of the null hypothesis are presented later in the research; each of these hypotheses isolates a set of variables by asserting that observed values for signal availability (0 or 1) are consistent with random distribution.

- *Null Hypothesis:* H_0 is the *Null Hypothesis*. H_0: Performing navigation using multiple technologies does not improve at least one of these conditions at any time: availability, speed, or inclusiveness.
- *Alternate Hypothesis:* H_a is the *Alternate Hypothesis*. H_a: Performing navigation using multiple technologies will improve at least one of these conditions at any time; availability of sensor based on location, speed of navigation, inclu-

siveness of systems (i.e. outside, inside, elevation, etc.) as the system moves autonomously through individual systems as appropriate.

The distributions and measures to evaluate can be seen in Table 1.1. These include a binary range of the system's availability, the assigned speed of the sensor system available, and the percentage of the total run in which the sensor system is available.

Table 1.1: *Distributions to Evaluate Expected Results*

	Navigation of System	Availability of Sensor (0 or 1)	Speed of Navigation (1 - 10)	Inclusiveness of Systems (Percentage)
Distribution 1	GPS			
Distribution 2	GPS,Wifi			
Distribution 3	Wifi			
Distribution 4	GPS, RFID			
Distribution 5	RFID			
Distribution 6	GPS, Wifi, RFID			

1.3 Definable Goals

The definable goals for this research include implementing prototypes for testing, and discussing the connection between the prototypes and models. These are:

- Examine use of agent-based model

  Discuss characteristics and applicability

- Formalize theoretical description of system-of-systems

Support why SoS is appropriate for a framework

Discuss independent systems and characteristics

- Perform agent-based modeling and simulation

  Perform simulations

  Collect data

  Analyze data

- Implement prototypes for testing (P_i)

  P_1: Path-planning on the Google Nexus S

  P_2: Path-planning on the Apple iPhone

  P_3: Matlab agent-based models

1.4 Organization

This chapter has clarified why the research was undertaken as a study of system-of-systems and set forth definable goals for the successful completion of the research.

Chapter 2 discusses the interdisciplinary aspect of the research which includes mobile devices, localization and path-planning, complex systems, modeling and simulation. This review selects seminal and state-of-the-art literature from the volume of literature available.

Chapter 3 describes the predecessor and final prototype. It presents the design and implementation of the device in light of the test physical environment. It also discusses the parameters, factors and independent variables as the experimental effort moves forward.

Chapter 4 introduces the methodology of the project, which is agent-based and utilizes both simulation and prototyping. The chapter offers information on agent-based modeling and formal specifications are noted.

Chapter 5 presents a detailed examination of the experiment and resulting data. It addresses the appropriateness of the design and any weaknesses observed.

Chapter 6 places the research presented in the context of possible real-world applications which are viable in light of the experiments presented. The conclusion also suggests specific avenues for future work.

CHAPTER 2. LITERATURE REVIEW

A rich and mature literature exists in a variety of areas. When Googled, the keyword *path-planning* returns a count of approximately two and one-half million articles while the keyword *simulation* returns over four million articles. The sheer number of papers in this interdisciplinary research requires sampling. The literature involves many disciplines and continues to evolve rapidly.

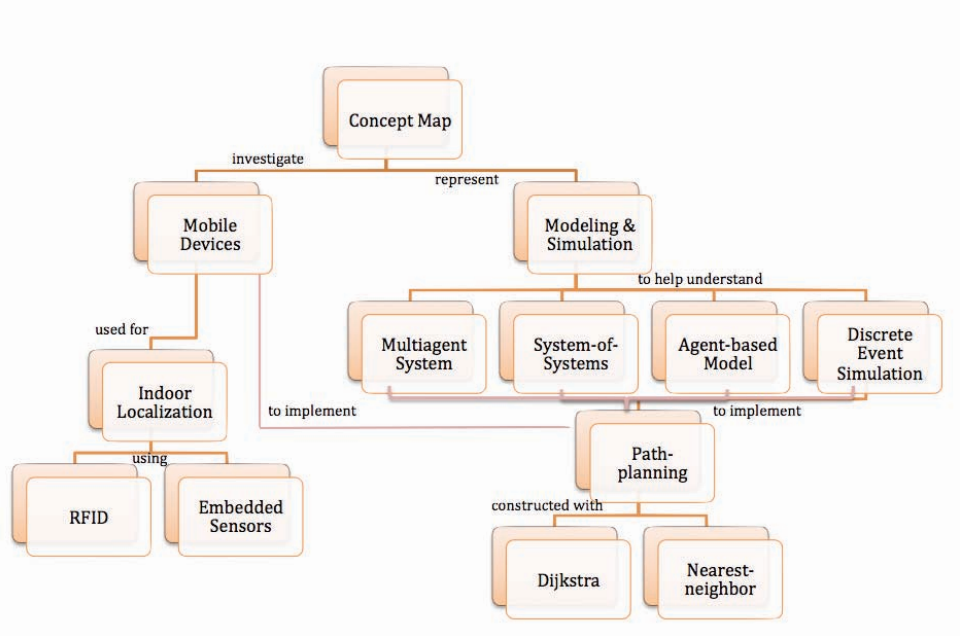


Figure 2.1.: Broad Concept Map of Literature Topics

For this research a broad category of literature is reviewed. A hierarchical concept map is provided in Fig: 2.1. The top level topics are mobile devices and modeling and simulation. Mobile devices are included as application prototypes (P_1 and P_2 described in Chapter 4) and will be implemented on two individual smartphones. From this category, the relevant and complementary portion includes indoor

localization either through RFID or mobile device embedded sensors. The specific applications reviewed in these papers are multi-topical (e.g. mobile robots, visual assistance, inventory management, etc.).

Modeling and simulation are explored through a variety of topics: multi-agent systems; system-of-systems; agent-based model; and discrete event simulation. As the research involves a framework to fuse the data of multiple sensor systems, a variety of methods and systems are explored. Agent-based and multi-agent systems are explored as an appropriate model for simulation. System-of-systems is important to explore, as the architecture for the proposed model involves mergine multiple individual systems.

Additionally, navigation is impossible without a discussion of path-planning. For this research, Dijkstra's single-source shortest path and the Nearest Neighbor algorithm are reviewed. This review is so broad, as this research encompasses more than one area. For this reason, any reviewed research that is multi-topic in nature, is contained in the section that is the most relevant.

A detailed concept map is presented in Fig: 2.2.

2.1 Mobile Devices

As mobile devices have become a ubiquitous technology, the need to compare and contrast various systems was required. As embedded sensors and prototypes are used in the research, the category of smart-phone was selected as the interface. In North America, deployment of mobile applications generally occurs on one of the top four mobile operating systems (Blackberry, iPhone, Android and Windows Mobile.) In addition to the sensors of the phone, the details of the environment were relevant to this research. As such, details of related work are shown below. To determine which platform would be the most suitable for the prototypes required two different but coordinated approaches:

1. determine the relative advantages and limitations of mobile device models;

Figure 2.2.: Broad Concept Map of Literature Topics

2. determine the capacity of a given mobile device to exploit RFID technology for the purpose of enhancing navigation.

The potential of Apple's iPhone against Google's Android has been compared in the literature and the results re-examined in this research. Take the openness of an operating system for application development. While there *are* many Java programmers, the iPhone OS is built on the long-running and stable Mac OS platform. Therefore the claim by Hall that few developers have experience with it, is spurious [4]. It has been suggested by Lin et al. [5] that market-share is a key factor in determining the ease of application development. In contrast, Oliver [6] analyzed the operating systems of selected phones with regard to the research goals of the devel-

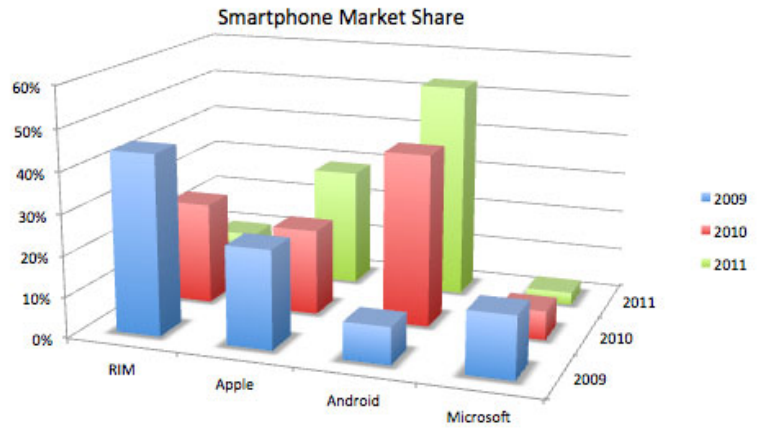


Figure 2.3.: Market Share of the top four smart-phones [3]

oper. Oliver's article provides a detailed background to the development challenges that the researcher faced in writing an application in Java for the Android. This literature review (and its selected topics) may be regarded as an update of topics addressed in Oliver's work, given the more detailed experience with mobile devices and their relative advantages and limitations.

Independent of literature in the field, a determination through experimentation with the iPhone presented significant hurdles. Problems with the iPhone as a development platform did not cause the research to move forward on the Android, but rather a deficiency which many researchers have discovered in any embedded sensor: noisy sensors with squared integration errors [7] [8] [9], for which no fix has been developed. However, iPhone testing of RF sensors was an essential step forward as this work allowed the research to isolate the problems and challenges of obtaining incoming signals, both native and programmed, as information to be fused with Wifi and GPS signals. This preceded the conclusion by Anvaari, [10], that between Android and iPhone, Android was the most open platform for development.

Table 2.1: *Characteristics of the top four smart-phone systems*

.	*Blackberry*	*iPhone*	*Android*	*Windows Phone*
Interface Selection	API available to determine network coverage	Framework available to determine network reliability	Class provided to monitor and return state of network	Class provided to select and receive network status
Bluetooth	Available, requires pairing procedure	Available, requires pairing procedure	API, scan and check for paired devices data transfer and connection management	Unavailable as of first version release
Background Processing	Service module available to send/receive background messages	Supported as of Version 4.0	Permitted, but any process can be killed to reclaim memory	Permitted for native applications. First version prohibits 3rd party control
Energy Monitoring	API's to check state of battery (e.g. level, charging, temperature)	Register selectors (msg service) re: battery state changes	Battery MGR class to detect status of battery	No API available to access this info

Continued on next page

Table 2.1: *(Continued) Characteristics of the top four smart-phone systems*

	Blackberry	*iPhone*	*Android*	*Windows Phone*
Power Saving Controls	Power saving is permitted-enable/disable screen, shutdown device	API query battery capacity, state, voltage, charging status	Fine-grained control over power consumption	No API available to access/control this info
Memory Management	JVM - allocate memory garbage collection, automatic paging	No garbage collection. Must use reference and retain count	Uses custom Java Dalvik VM. Ensures multiple instances can run efficiently	Managed code, includes built-in garbage collection
Persistent Storage	File system, SQLite DB, Persistent Store	CoreData, FW optimized XML format	File I/O SQLite DB	File system access prohibited. Isolated and stable, for each process
Location Sensing	GPS- internal/external receiver, Geo-location	CoreLocation framework uses hw to find location	GPS localization and cell tower triangulation	API to use input from GPS or Wifi

Continued on next page

Table 2.1: *(Continued) Characteristics of the top four smart-phone systems*

	Blackberry	*iPhone*	*Android*	*Windows Phone*
Development Language	Java - platform independent	Objective-C, C MVC design pattern	Java - platform independent	Currently only C Sharp
Development Environment	Java development kit, Eclipse plug-in	iPhone SDK $99.00 / year	Android SDK, Eclipse plug-in	Visual Studio Express for Silverlight SNA
Code Signing	Signing Authority Tool public/private key cryptography	Signing Certificates	Signing Certificates may be self-signed	Certification testing prior to code signing
Testing, Emulation and Tools	Emulators available for download for all models	Large number of tools available	Large number of tools available	Emulator, SNA Game Studio, Expression Blend
Application Deployment	BB App World, vendor guidelines, on-line submission	App store, must be accepted by Apple for design and content	Android Market, $25.00 registration fee and agree to Dist agreement	Phone Marketplace $99.00 annual fee, and online certification/submission

Continued on next page

Table 2.1: *(Continued) Characteristics of the top four smart-phone systems*

	Blackberry	*iPhone*	*Android*	*Windows Phone*
Hardware	OEM Blackberry devices	OEM iPhone, iTouch and iPad	Multiple hardware vendors	Multiple hardware vendors minimum system requirements

2.1.1 Discussion of Mobile Devices and their Environment

A summary of the characteristics of each system is presented in Table: 2.1. A lengthier discussion of background information is included in Appendix A. Mobile devices are continually adapting and reinventing the state of the art. At the time of the prototyping, the Apple iPhone and Google Nexus S were selected as they both enjoy the largest market share and each had specific hardware and software features necessary for the study.

2.2 Indoor Localization

Indoor localization is a topic area that is oft-researched. This section includes work in the areas of RFID localization and embedded sensors on a mobile device.

2.2.1 RFID

Although radio frequency identification (RFID) technology is not new, innovative uses for it continue to grow. It has been used in topics that include inventory management, smart card technology, robots and localization. In the context of this

research, the implementation of RFID is used to assist in determining location. Indoor localization via RFID technology presents a natural avenue for research and development. For example, brute force saturation has been used: the researchers embed RFID sensors with minimal information stored into their memory (tags) in a physical environment in a grid format. (Tagging is thus a term used to indicate the deployment of RFID tags.) At this point it is important to note that implanting fixed rigid tags in a grid is a significant step away from imitating the natural motion of a human being, whose movements are guided by doors, stairwells, elevators and other familiar landmarks.

Kim and Hong installed grids of RFID tags on the ceiling of an interior space; each tag was programmed with coordinates and transmitted this information to a device (not necessarily a smart-phone) which had a database of information to localize the device [11]. The programming relied on a determination algorithm which used the signal obtained from line-of-sight measurements. The algorithm deleted the outliers which allowed it to extract the target location by calculating the mean value of the remaining data set.

The limitations of this approach are obvious, and have been demonstrated in other papers. Take passive UHF RFID tags in a grid format in a room: the algorithm created relied on the Angle-of-Arrival to determine the users location. The conclusion was that RFID alone was not enough for effective indoor localization, but would be better combined with other technologies for higher accuracy. [12]. The essence of the approach was that a single mobile tag, attached to a computing device, could determine the user's location; even with pre-arranged grids and coding, localization with RFID alone could not be rendered a reliable advance.

The history of RFID technology in indoor localization can be traced through the works of Zhao [13] who, in 2007, published their first paper on a team experiment with fixed grids; to overcome the obstacles they observed and which appear in the literature, the team stored a proximity map on the reader and an elimination algorithm to calculate location. The results brought the experimenter to within .14

meters of the true location; however, even these results, documented in their follow-up papers [14] and [15], demonstrate the likelihood that massive grids will never make RFID a stand-alone enhancement technology. These results were confirmed by Pradhan et al. [16], who conducted field tests under real operating conditions with ultra high frequency RFID tags placed at fixed interior locations, while the user carried the reader inside the building. As with other research, mapping of the environment was a prerequisite to localization and was done by measuring the signal strength of the RFID tags from different locations; this team obtained errors of over ten meters from the true location.

Zhu, Wei and Hu [17] compared LANDMARC and VIRE. LANDMARC was discussed in a previous paper by Ni et al. [18]. The authors were interested in the performance of the positioning algorithms in active RFID systems. An improvement was presented as a reduction algorithm for higher accuracy.

An interesting step forward was the utilization of two signals, active RFID and ultrasonic, proposed by Yuan et al. [19]. The research gathered signals to localize a user indoors, and then perform path-planning from a starting point to a final destination. The first method used was range measurement: the local 'landmark' would send out both an RF and Ultrasonic signal, and the user's terminal would receive these independently. As RF travels faster then Ultrasonic, the discrepancy was used to measure the Time-Difference-of-Arrival (TDOA) between the signals.

The second method involved using A* to perform path-planning using an occupancy grid. Putting the limitations of fixed grid-based installations to one side, Yuan advanced the research by exploiting two signals and deploying the information for localization and path-planning purposes.

In this regard, it is worth noting that a significant branch of the literature assumes that indoor navigation systems are designed to guide mobile robots. As far back as 2003, Hahnel et al. [20] proposed a fusion of data from lasers with RFID to improve localization of mobile robots or persons. Conversely, a study investigating the potential of stand-alone and purpose-built computers was explored by Yelamarthi

et al. [21]. They described a Smart-Robot system with an integrated navigation system using RFID and GPS. The system was designed to help a visually-impaired user navigate to a location through a set of predefined routes. The smart-robot also avoided obstacles using ultrasonic and infrared sensor inputs. The conclusion reached was that RFID and GPS in combination for navigation assistance was technically and economically feasible. The Smart-Robot system could potentially improve the quality of life for the visually impaired by making routine tasks simple and feasible.

BlindAid, a project presented by Mau et al. [22] lacked presentation of any visual aid, such as a Google Map. Instead audio cues were used for the sight-impaired. The research was designed using passive RFID tags deployed within buildings and equipping the software to determine their location. Some of the methods for this work were map generation by using a floor-plan and noting the location of the RFID tags on it. Path-planning was performed using Dijkstra's algorithm to provide a path for the user to follow. An early Dell Axim phone was used running Microsoft Windows 5.0. The software was written in C# .

It is worth noting that BlindAid was clearly an improvement over the computer in a fanny-pack, which supplied feedback with vibrations to orient the user. Willis and Helal [23] proposed an RFID infrastructure; shoe and white-cane integrated reader; blue-tooth connection, and RFID tags with spatial information. This infrastructure was used to offer a navigation system to blind users on a college campus. The localization was performed using pre-mapped rooms and paths and then applying the spatial coordinates from the passive RFID tags.

The specialized requirements of fire-fighters, are an interesting example of research that tests the limitations of a smart-phone in dangerous environments. Assuming passive tags, densely located in a building, an Android HTC Droid phone was programmed in Java (along with an RFID reader in a glove) to signal the fire-fighter with cues for the physical layout of the building. A textual floor-plan was included and transmitted, via cues, to the phone.

The benefits of the system were listed as: low deployment costs, scalability, on-demand information and portability. However, the pre-planning is considerable: densely embedded RFID tags, pre-mapping of the RFID and floor-plans in text format; streaming video to overlap mapped environmental information; and a glove which the user was obliged to place directly on the RFID sensor.

Other specialized environments have also been investigated. In addition to deploying an RFID infrastructure, Ahmed [24] created middle-ware dubbed *Guardian Angel* which allowed improved care-giver monitoring in an assisted living environment. Although localization accuracy was important to the study, the system was designed with privacy in mind. This allowed the user to control what information about their location could be monitored by health care givers.

A summary of these papers, including the methods and outcomes is included in Table: 2.2.

Table 2.2: *Characteristics using RFID*

Research Outcome	Research Methods	Author
Indoor Localization passive RFID plus Wifi	Localization - line of sight custom algorithm	[11]
Localization algorithm passive UHF RFID tags	Localization - Angle of Arrival Tags placed in grid	[12]
Indoor Localization active RFID	Localization algorithm RSSI signal measure interpolation and elimination	[13]
Indoor localization active UHF RFID tags mounted tags	Map generation - using measure signal strength hypothesis testing and k-Nearest Neighbor	[16]
Comparison of previous methods	Localization - using RSSI Measure previous performance	[17]

Continued on next page

Table 2.2: *(Continued) Characteristics using RFID*

Research Outcome	Research Methods	Author
Localization and Navigation	Localization - RF & ultrasonic signals - time difference of arrival, Newton iterative process	[19]
Data fusion- RFID + lasers Localization using probabilistic measures	Pre-existing map Monte Carlo for localization	[20]
Wayfinding for blind users, active RFID	Map generation to localize. Path-planning using Dijkstra algorithm	[22]
Localization, Navigation obstacle avoidance using a smart robot for blind users	Indoor & outdoor localization. Navigation algorithm using pre-defined routes Sensor use for obstacle avoidance	[21]
Wayfind for blind users with passive RFID and reader integrated white cane	Pre-mapping of rooms and paths. Localization using spatial coordinates	[23]
Accurate localization plus navigation for emergency workers or blind users	Passive RFID + audio landmarks. Back-end processing, Android smart-phone	[25]
Passive RFID for localization	Localization with user control for privacy control	[24]

2.2.2 Embedded Sensors

Due to the popularity and mass distribution of mobile phones,location-based services have been a natural progression for the platform. As the previous discussions suggest, advances in localization services are a function of;

- capturing and manipulating additional information, from any source; and

- fusing the information into data that can be projected into a user-friendly format.

The gold-standard in location-based services is Google Maps. This research utilizes and extends this as described in Chapter 3.

The architecture of the mobile phone took a dramatic step forward with the deployment of MEMS (Micro Electric Mechanical System) technology. Chips installed on the motherboard of these devices offer information via an Accelerometer, Gyroscope or Magnetometer (Electronic Compass).

The familiar pattern of new technology having difficulties achieving reliable location data now repeats itself. The challenge is how to use these devices effectively to provide an accurate solution. Some of the prototypes developed in this research have involved mobile platforms and their embedded sensors.

Several works included the idea that accelerometer data was easily obtainable from ubiquitous mobile devices. The research was done believing accelerometer information was suitable to augment other systems (either to improve accuracy or conserve battery.) In Constandache et al. [26] the team describes deployment of an accelerometer and digital compass from a mobile phone; the authors measured the walking speed and orientation of the user. They drew data points from the mobile device's sensors, as they were more energy efficient than GPS and Wifi based systems. The drawback was the level of accuracy, which averaged over ten meters. The authors concluded "noisy phone sensors and complicated human movements present practical research challenges. To overcome the deficiency, the authors proposed recording walking patters. The MEMS-supplied data could not be reliably digested by the hardware and software commanded by the investigators.

To continue isolating the information of the MEMS-based sensors Hsu and Yu [27] theorized that coarse indoor localization could be maintained through accelerometer data alone; their approach computed an object's displacement by transforming the information gathered by the accelerometer. Their paper presented a theoretical and simulated version of accelerometer-based positioning. The conclusion

points to the compounding error occasioned by successive integrations, i.e. the error exponentially increases as the user attempts to navigate.

Another group categorized various states of user activity from data emitted from an Android NTCG1. Specifically, accelerometer data was gathered to classify the activity. Parnandi et al. [28] combined this data with the last known GPS coordinates when the user moved indoors to provide a coarse indoor location of the user. The following method was employed: the GPS coordinates for the last known outdoor location was logged; once the user moved inside, accelerometer data was logged to a file at specific intervals. It was the user's hesitation that triggered the change in state which the software was programmed to interpret. Once the user stops moving, the changed states caused the system to attempt to calculate the current location of the user based on the logged accelerometer data.

The authors performed case studies using both naive Bayes and dynamic time warping strategies to classify the user's activities: standing still; walking; going up or down stairs; or going up or down the elevator. While the dynamic time warping was found to be more accurate, the computational complexity was such that the increased accuracy was not enough benefit to select it. It is worth noting that the authors illustrated the required placement of the mobile device as strapped to the users foot.

Combining Wifi and accelerometer readings from a mobile phone Ofstad et al. [29] theorized that user localization on a Google Map could be supplied via mobile phone (Nokia N95), which directed the user to one of three desired locations. In the system as modeled, the time interval for a sensor recording was set to one second. At one minute; a filter classifies and records the user's current activity based on the recorded sensor data, i.e. sitting or standing. The limitation of this work was the lack of information displayed advising the user of their current location and there was no investigation into combinating or fusing signals.

Fusing three sensor technologies and an algorithm to enhance the positioning performance for first responders was addressed by Amanatiadis et al. [30]. Their pre-

liminary experiments showed an improvement in navigational accuracy derived from exploitation of RFID, accelerometer, and image processing. Their indoor navigation system, however, was based on sensor data from first responders wearable modules. The researchers required an additional input device to supplement the input from the smart-phone.

A rather odd experiment: data from accelerometers, gyrometers, magnetometers and barometers were evaluated through a device created by Kourogi et al. [31]. A PDR (Pedestrian Dead Reckoning) Evaluation Kit was developed; PDR software and self-contained sensor module enabled indoor and outdoor localization systems. At the time of publication the hardware and software were available for evaluation from SHIBUYA KOGYO for around $5,000.00 No citations (other than the authors) suggest that this became commercially viable.

Testing the capabilities of an unaugmented smart-phone, the research effort by Dekel and Shiller [32] attempted to exploit an iPhone 3GS. In this case accelerometer and magnetometer data were gathered. The approach counted steps multiplied by estimated step-size to supply localization data. The results required the user to supply user gait characteristics; an improvement from strapping the phone to the user's foot, which other studies require. In the work by Hynes et al. [33] investigator's fascination with footwear-fixed inputs may be tracked. This study relied on recorded accelerometer data to analyze the user gait. The study used a low pass filter to detect periods in the data. This allowed the researchers to determine between two states dubbed walking and non-walking. The advance in analyzing gait and activity analysis has not supported any further advances in navigation enhancement technology.

While Liu et al. [34] used the embedded accelerometer to determine the periodic pattern of each step, what is of more interest was the deployment of the Weibull function, used to represent the wireless signal strength distribution over time. To be effective, a database of Wifi signal strength values must be measured and recorded to a database in order to manipulate the information received in its raw form. Liu et al., combined this data with data derived from the pedestrian movement, via the Nokia

N95s accelerometer. This permitted the authors to combine Wifi-aided positioning with context detection, i.e. whether the movement detected was static, walking slowly or walking fast.

2.2.3 Discussion on Indoor Localization

There is active research into methods for better indoor localization. The criteria employed for including a review in this research is that the work utilized RFID sensors or embedded mobile sensors (e.g. accelerometers, gyroscopes or magnetometers) or a fusion of sensors and localization techniques. No single solution has been found to successfully solve indoor localization or navigation. A summary of the review is included in Table: 2.3.

Table 2.3: *Characteristics using embedded sensors*

Research Outcome	Research Methods	Author
Indoor localization based on steps and path once GPS fails.	Map generation - manual process. Signature matching for localization	[26]
Indoor navigation sensor fusion: IMU, RFID and digital camera	Classification of activity Measurement of vertical acc Fuzzy algorithm for fusion	[30]
Course indoor localization based on accelerometer measures	Simulation of localization using accelerometer - object displacement through info transformation	[27]
Course indoor localization using accelerometer data from a smartphone	Accelerometer plus GPS Classification with naive Bayes and dynamic time warping	[28]
Mobile phone accelerometer measures	Categorization of user context based on accelerometer readings	[29]
Commercial Application for localization	Calculations based on sensor attitude, velocity and movement	[31]

Continued on next page

Table 2.3: *(Continued) Characteristics using embedded sensors*

Research Outcome	Research Methods	Author
Indoor navigation using smartphones	Calculation of dead reckoning based on pedometer activity	[32]
Indoor navigation using smartphone	Calculation based on acc measure. Classification algorithm for gait, activity	[33]
Indoor navigation using acc and Wifi	Measure signal with Weibull function. Accelerometer measure. hMM particle filter to combine	[34]

2.3 Modeling and Simulation

In addition to application prototypes, this research also implements a model and simulation. Modeling is the abstraction of a concept to allow the development of a formal specification of the system. Simulation is the computerized execution of the model over time to study the interactions of the agents. The concepts of simulation and Agent-Based Modeling (ABM) are presented.

2.3.1 Agent-Based Modeling

The search to replicate, a series of patterned events began with John von Neumann's design for machine-to-machine replication, he followed a detailed set of instructions [35]. His colleague at Los Alamos, Stanislaw Ulam, (also a mathematician), proposed an automaton which could express itself in a chess-board like grid. [36].

Another step forward came with the work of John Conway; in 1970 he constructed the 'Game of Life'; in which each cell has its own preprogrammed behavior and, once launched, the cells (or agents) will interact perpetually according to the

rules. The simulation appears on a two-dimensional checkerboard [37]. Although Schelling's initial 1971 model did not use computers, the agents he described (coins on graph paper) were autonomous and interacted with observable and emergent outcomes [38]. The design of these simulations was to capture human emotion as the motive which explained change of state in each agent.

Many of the advances in modeling produced computer code which simulated small communities. In Epstein and Axtell's Sugarscape presented the modeler with a game-type interface in which the wants and fears of agents were exploited. Once launched, the scenario would play out on the screen for the players edification [39].

From the perspective of this research, Parunak et al [40] offered a case-study of vehicle transportation networks; the team compared modeling with agents versus mathematic modeling applicable to the movement of product through the network. Similarities and differences were presented as well as criteria for selecting one approach over the other. At the time, ABM was a relatively new approach to system modeling and simulation. The conclusion described ABM as "appropriate for domains characterized by a high degree of localization and distribution [of decision-making]. As focus on improved agent navigation would exploit ABM's advantages, rather than focusing on the dynamics of movements explained by physical laws.

The development of the field proceeded rapidly with the deployment of ever-increasing computer resources. Closer in time to the current research, Macal and North published their "Tutorials on Agent-Based Modeling", part I in 2005 and part II in 2006 [41] [42]. Identification of the variables, relevant definitions, an example of the model and its simulation, and interpretation and analysis of the results were supplied by the authors. The authors compared the advantages of ABM to conventional simulation approaches.

In their research, a MatLab simulation was written on the assumption that the navigator was independent, but would want to be aware of the choice of available signals, information pertinent to choices in path-planning, and, most importantly, the cost of errors in path-planning. Charles Macal and Michael North's ABM obliges

the code writer to make these features of the navigator explicit, which this research does, even if ABM does not oblige the code writer to expand the types of agents and render the owner of the physical environment an agent in the simulation whose preferences and costs must be taken into account. That latter step, as is explained in other sections of this research, is reserved for future investigation.

The deployment results of modeling and simulation was addressed in Bauer, Beauchemin and Perelson [43] who reviewed various agent-based models relevant to host-pathogen systems, and noted limitations and challenges of agent-based models. This research approached modeling only *after* deployment of two prototypes (one abandoned, given the known accelerometer deficiencies). The modeling effort pointed to refining the Java-based manipulation of multiple incoming signals. The development, in this research, of the Java programming (for the Android) and the MatLab programming (for ABM simulation) went hand-in-hand, the preferred approach suggested by Bauer, et al. who posit that modelers should quantitatively validate the results of their simulations with independent experiments or with reports in the literature.

Norths work on a team at the University of Chicago, lead directly to the development of the Repast modeling toolkit, which facilitate modeling and simulation [44]. Getchell [45] reviewed theory and practice of agent-based modeling and evaluated toolkits, including RepastPy, Repast Symphony, Breve and described his experience with these programs to develop increasingly sophisticated ABMs. Noted was a greater-than-anticipated learning curve in using Repast Simphony, although the author determined it was the best for network and grid models with visualization of non-spatial temporal data. The author concluded that RepastPy was the best for prototyping of network or grid models. This modeling and simulation was done using MatLab. It is designed to handle the mathematics associated with an extensive grid-based representation. In 2009, the article "Tools of the Trade" by Nikolai [46] cited MatLab among the fifty-three programs available for modeling and simulation.

MatLab's subspecialty is considered matrix-based math, computation and computer simulation.

> "MatLab is a high-level language that includes matrix-based data structures, its own internal data types, an extensive catalog of functions, an environment in which to develop your own functions and scripts, the ability to import and export to many types of data files, object-oriented programming capabilities, and interfaces to external technologies such as COM, Java, programs written in C and Fortran, and serial port devices."

In preparation for this review, the work of Brian Heath [47] was reviewed. Heath reviewed 279 articles in the literature; of interest, he diagrammed a melding of best features to serve as the medium between the real world system and the simulation model, a technique he called CM4S. His case study involved naval warfare (WWII) in the Bay of Biscay, featuring anti-submarine warfare. In distinction, however, the preferred approach for the development of any computer application is the deployment of the prototype in the field, with modeling and simulation as the discipline to improve real world results rather than use of a medium as the authors research suggests.

An example of this preferred approach, is the simulation designed by Zhao and Li [15] which involved agent-based modeling, path-planning and driver behavior for a traffic simulation study. Typically, each agent in the prototype was bound by a number of behavioral rules; memory, learning and adaptation modules were resourced by the agents; deficiencies such as limited memory were included. In general, both the route planning process and driving process were simulated which allowed the modelers to create agents who behave in realistic fashion, even if the shortest path was not their choice. Zhao's team demonstrated that coherence with the failings of human behavior, at least as to modeling path-planning, is a measure of the model's success; the enhancement which the team offered drivers took into account known human tendencies expressed in sometimes irrational behavior in the driving experience.

2.3.2 Simulation

Discrete event system specification (DEVS) was introduced by Bernard Zeigler in the early 1970's [48]. DEVS is a formal specification over discrete time events that is both modular and hierarchical. In the context of this research, the atomic model is used. The atomic model has inputs and output with transitional states linking them. The advantages of making DEVS's specifications explicit are explained more fully in the methodology. In brief, from the modeler's point of view DEVS permits the modeling effort to more closely mirror the thoughtprocess of the researcher. This is accomplished by dividing research into logical phases. Time is not ignored; it is accumulated into the states the DEVS specifications call for.

DEVS also enables more conservative use of computer resources, as events are logged by changes in state instead of time. This allows an agent to be tracked in more detail and stochastic events programmed into the code.

The formal model of DEVS combined with Z formalism was proposed by Traore [49]. The combined formalisms enabled rigorous analysis of models and their properties. A study of urban bus transport was introduced. The author combined this one-to-one linking to determine a formal analysis using the newly minted Z/EVES. This demonstrated that any agent-based model meeting minimal formal requirements can be linked to a discrete event model. According to Onggo [50] the advantage of this approach is the modeler's do not need to change their modeling paradigm.

A case study modeled fishing activities complete with a harbor and fixed harvest zones, replicating the real-life dense management of off-shore fisheries. The model was translated into DEVS specifications; in some detail, Duboz et al. [51] investigated a multi-agent simulation as the agents paths were traced as discrete events dividing known states, rather than by exhaustion of time-steps through virtual grids.

Writing a JAVA application for the Nexus smart-phone enabled this mobile device to fuse different inputs to enhance pedestrian navigation. A formal demonstration of the benefits of exploiting DEVS specifications appears in Gianni [52]. The

opportunity to call for packaged code via the Java Database Engine (JADE) was disciplined "by a formulation of DEVS in terms of a multi-agent simulation in terms of a software [the team named] simJADE." The framework was used to simulate a variety of emergency scenarios. The authors tested and validated their design through repeated simulations.

Preparatory research for navigation simulations in MatLab has been explored in the agricultural context. Bochtis et al. [53] converted a topological map to a two-dimensional grid on which the defined states (i.e. obstacle, start state, free or goal state) could be mapped. The path-planning occurred by using a graph search algorithm to find the shortest path. The researchers performed simulations of automated path-planning. Although the technology was developed for agriculture, and specifically seed and harvest machinery, the exploitation of a topological resource (in this case a map of the Purdue University campus) is viable.

The work of Usher and Strawderman [54], investigated states. Pedestrian behavior was selected based on a literature review and narrowed down to collision avoidance, changes in speed and trajectory, passing strategies and distance between objects. The simulation was found to be comparable to the empirical experiment for displaying navigation and pedestrian behaviors. The programming in C++ yielded a model rather than an enhanced navigation tool.

Although the work by Wei et al. [55] was not useful for the current project, the ideas were interesting enough for inclusion toward future work. The authors created a framework for hybrid modeling, in effect a fusion of models. The goal was to combine various simulation models into a single model capable of complex dynamics. The authors combined integrated agent-based models of different scope and scale: computational microscopic model for individual pedestrians; macroscopic model for crowd movement, and an agent-based model for vehicles on the road. The models were connected through both direct and indirect data and event flows. The infrastructure was validated through experimentation and will allow future research to combine many types and scales of scenarios.

2.3.3 Discussion of Modeling and Simulation

Like previous background topics, modeling and simulation is a broad topic. The review was bound by specific, relevant and recent works. The simulation section was limited to works in the area of discrete-event simulation (DEVS) and pedestrian simulations. Modeling was constrained to agent-based modeling for a variety of purposes. A summary can be seen in Table: 2.4.

Table 2.4: *Table summarizing Modeling and Simulation*

Research Outcome	Research Methods	Author
Simulating pedestrian navigation with human behavior	Pedestrian navigation while accounting for collision avoidance, change in speed or trajectory, passing and distance between objects	[54]
Multi-agent model combining multiple formalisms	Generate DEVS model, Combine with other DEVS models, Create localization grid as a cell-DEVS	[51]
Path-planning using a discrete transition graph and graph search algorithm	Map generation - convert topological map to 2D grid cell, Path-planning - use graph algorithm to find shortest path	[53]
Fusion of models for complex simulations	Implement infrastructure to combine models. Validate through experimentation	[53]
Framework for agent-based DEVS for emergency simulations	Create/extend a framework for disaster and emergency prepareedness. Validate through repeated simulations	[52]

Continued on next page

Table 2.4: *(Continued) Table summarizing Modeling and Simulation*

Research Outcome	Research Methods	Author
Agent-based model for simulation of driver behavior	Create model with rules about memory, traffic info, character, route planning, and driving. Navigation: Dijkstra	[15]
Review of ABM	Review ABM, note limitations and challenges, suggest verification approaches	[43]
Review of theory and practice of ABM	Evaluate existing ABM toolkits, RepastPy, Simphony, and Breve	[45]
Introduce and perform ABM as a simulation approach	Identify data and variables. Define and implement a model. Perform simulation. Interpret and analyze the results	[42]
Historical review of ABM and applicability to complex system	Conduct lit review and evaluate results. Provide historical and philosophical review of ABM	[47]
Compare ABM to mathematical equations	Determine the appropriate situation for ABM v mathematical modeling	[40]

2.4 Complex Systems

Research in multi-agent and system-of-systems has, relevant to this research, focused on allocating the various navigation capacities and responsibilities between human navigators needing an enhanced experience and a number of mediating agents who acquire information, manipulate it, and supply the enhancements needed or desired by the human navigators. A variety of theoretical and practical challenges mark the research in this area, as mediating agents are defined and deployed.

2.4.1 Multi-agent System

Multi-agent modeling challenges the investigator to move beyond defining and launching models to simulate human navigators with such deficits as old age, disabilities, confined physical environment, etc. The architecture by Falco et al. [56] includes specialized agents responsible for path-planning, path-building, user-orientating, tracking user-position and locating the user on the map. The incoming signals are manipulated and broadcast to the navigators according to their precise needs, given their current state. The authors suggested a range of means by which communication to the navigators could be achieved; however, the navigators, reduced to a role, makes the system too disconnected from the navigator's own needs.

Mediating agents have been explored, one trajectory for the research in the last decade. The navigator of an intelligent wheelchair would be able to select the type of control it should use to allow the user to navigate more easily, precisely, and safely. Y. Morre and A. Pruski [57] included functionality by allowing the system to record previous routes traveled to give the agents intelligent choices based on a hidden Markov Model and tuning parameters. If a route was recognized, the system could suggest the next move based on prior performance. Avoidance agents, wall-following agents, right- and left-turning agents, etc., were operational for limited trials in the wheelchair itself. The agents were modeled in multi-agent simulations before field work began.

The background that preceded these two instances of research exploiting multi-agent modeling, as applied in the field, is reviewed. A multi-agent system is typically composed of some number of multiple, intelligent agents as consumers of computing resources. Typical characteristics of the agents in the system are autonomy, awareness of surroundings limited to a defined purview, and decentralization of computing power [58].

The opportunity to endow agents with intelligent capacity has attracted notable theoretical and practical research efforts. The potential offered by a division

of labor among agents suggests that, from a modeling point of view, any number of intelligent, non-human agents may be assigned targeted roles and the ability to communicate; that is to learn, remember and adapt to the significance of information. Additionally, the agents are able to provide information to others so that each agent has more complete information.

Russell and Norvigs seminal work, "Artificial Intelligence: A Modern Approach", now on its third version, offered a comprehensive review of the theory and practice of designing and deploying artificial intelligence in task management. "A truly autonomous intelligent agent should be able to operate successfully in a wide variety of environments, given sufficient time to adapt" [59]. The goals of multi-agent systems were detailed as accessibility, determinism, dynamics, discreteness, episodicity and dimensionality; each of these goals are considered in both the Java programming for the Nexus and the MatLab programming for the various simulations, as explained here. In this regard, the work of Gerhard Weiss, who edited "Multi-Agent Systems, A modern approach to distributed artificial intelligence", is relevant [60].

The paper by Tao and Huang [61] reported on integrating an innovative environmental information sensing technology with multi-agent systems, to enhance the ability of traffic simulation systems. Along the way they developed a traffic simulation system titled JADE, an agent-based framework in Java.

A methodological approach for the developer was presented by Abdelaziz, Elammari and Branki, Multi-Agent System Development (MASD) methodology [62]. A virtual car rental operation illustrated the point-by-point construction of a multi-agent system. The team focused on a design methodology to assist multi-agent system designers through software development life-cycle, from system requirements through working code.

The principal strengths of the methodology was based on three important phases: concepts, models, and process. MASD provided extensive guidance for the process of developing and communicating design within a group. The traceability of the design, as each step progressed, allowed the developer to be satisfied that the

design was "justified and validated by the methodology [62]. This is overly optimistic, given the realities of a system that functions as a human computer interface on one hand, and models the view in a simulation of the real environment.

Code-writing is work-in-progress: what disciplines the developer is the real-time experience of transition from one organizational perspective to the other and back again. In virtually all cases, trial and error will trump elaborate methodology. Nevertheless it is worthwhile, for a developer to know that some theoretical attention is being given to an organized process.

The analysis of Bernon et al. [63] was more practical. ADELFE offered the developer guidance. Described as a multi-agent oriented methodology suited to adaptive multi-agent systems, it is better seen as an editor which promotes best practices in achieving successful interaction among agents. The team illustrated the methodology in a case study of a timetable design.

The paper by An et al. [64] presented the design and implementation of negotiation agents that vied for acquisition of finite resources. The agents that negotiated did not know the reserve price of each resource and were allowed to de-commit from existing agreements. Their experimental results show it improved agent performance and achieved better results.

This idea has immediate application as, even if finances are deemed unlimited, resources are always insufficient to supply all the information that human and intelligent agents need. It is more useful to calculate virtual costs that agents pay when they drift off course, given path-planning as presented in topological format. In the case of this research, the negotiation occurs in the modeling as it assumes that finite resources require negotiation by the developer on behalf of the agents simulated.

As the discussion moves to the theoretical aspects of developing multi-agent systems, there is a need to review agents both in-field and via modeling. Multi-agent systems can be complex and elaborate. It is useful to view these agents as participants in an organization and, at that level of abstraction, to examine transition in the organization itself. Matson [65] distinguishes between an initial and reorganizational

processes. The research shows it is computationally better to begin with a small organization and grow it, reorganizing many times, rather than initially organize a very large organization in a single process. Follow-on research "Transition Process Distinction in Multi-Agent Organization", points the way to reorganizational challenges such as the owner of a physical environment as an agent [66]. Including the owner obliges the system designer (now a system-of-systems designer) to ask if the "set of available agents can satisfy the set of goals through playing the set of roles".

The fact that this research will pursue enhancement of the navigation experience by deploying the resources of intelligent agents forces the examination of the organization. Is a new participant or agent, whose interaction is modeled, able to focus on mistake and cost avoidance over successful deployment of a given technology. It is necessary for the modeler to show the agents as intelligent agents.

Organizational analysis, as a resource for developers, was presented by Argente et al. [67]; the team proposed an agent organization model based on four concepts: organizational unit, service, environment, and norm. The main features of an organization were characterized by structure, functionality, dynamics, environment and norms. The utility of the research suggests that goals, projected onto an organizational agent, are a means of focusing the attention of the developer, for whom transitional states of a single agent-in-motion are not the total goal of the research effort.

Wood and DeLoach, [68] developed a multi-agent system methodology (MaSE) based on the idea of agent classes and the communication between them. The methodology was an extension of object-oriented environment development. The method was broken into seven steps along a logical progression. In the context of this research a system-of-systems may be substituted for a single system, given that when moving from Java to MatLab, each effort is assignable to a different and, possibly disconnected, system.

The seven steps of the progression were:

- capturing goals:

(a) the programmer's experience of identifying the transition states for the navigator,

(b) the navigator's experience of visualizing the environment via the enhanced display of Google Maps, and

(c) the programmer's experience of simulating the agent's navigation experience;

- applying use cases: accomplished using the table of characteristics and relying on DEVS formal specifications;
- refining roles: the principal role assigned in the field was the navigator; in simulation the various intelligent agents were assigned roles, which include path-planning and information updating;
- creating agent classes: as MatLab is not an object-oriented language per-se, class definitions were developed and deployed in Java;
- constructing conversations: the research models conversations between agents in path-planning, updating the agent position and updating the navigation state (in MatLab);
- assembling agent classes: generating the agents and their behavior; and
- system design: the initial and transitional states of the agent as navigator were defined, as explained in the methodology section of DEVS formalism.

There were a number of goals of this methodology; support for automatic code generation using a MaSE tool; creation of a proven methodology; and guide development through design and implementation. A limitation of the method was that only one-to-one agent-interactions was supported. This is overcome in the research by using the Android as a platform. It processes the various intelligent agent and navigator inputs and organizes responses to them in real-time and on the fly.

There is a need to exploit multiple and overlapping communication paths; to address this need, Oyenan and DeLoach [69], aimed to implement an information system that could adjust its processing algorithms to provide required information at various levels of efficiency.

In this regard, a simulation of multi-agents that exchange information, in the work by Silva et al., [70], is relevant to future research which will seek to exploit learning by intelligent agents of environmental hazards and conditions generally. Path-planning based on Dijkstra's algorithm was coupled with information sharing; path updating based on dynamic information is to be exchanged among the agents.

To address limitations in multi-agent path-planning, Wang et al. [71] introduced MAPP, a tractable algorithm for multi-agent path-planning on grid maps. MAPP has low polynomial complexity in time, space, and solution quality. Using A* as an alternative to Dijkstra, the team created a tractable class of multi-agent path-planning problems, of which scalability and scarce computer resources stand out as deserving the most attention.

The authors believed this to be the first study that formalizes restrictions to maximize path-planning output while guarding against over-use of finite computer resources. This is necessarily a point to consider as the concept of public 'shared space comes under evaluation; multiple navigators each having access to multiple and available intelligent agents will present real-time issues in accessing and deploying computer resources.

A summary of multi-agent systems is included in Table: 2.5

Table 2.5: *Characteristics of Multi-agent Systems*

Research Outcome	Research Methods	Author
Multi-agent system for dynamic path-planning	Navigation performed by Dijkstra. Agents exchange info re dynamic environment changes for replanning	[70]

Continued on next page

Table 2.5: *(Continued) Characteristics of Multi-agent Systems*

Research Outcome	Research Methods	Author
Methodology for multi-agent system engineering.	Capture goals, apply use cases, refining roles, create agent classes, construct conversations, assembling agent classes, system design	[68]
Navigation assistance for the elderly	Create agents; path-planning, path building, user orientating, tracking and localization	[56]
Intelligent wheelchair navigation assistance for the elderly	Multi-agent system, intelligent agents, hidden Markov Model and tuning parameters for vector recognition	[57]
Methodology for multi-agent system development	Develop multi-agent system development methodology. Test with software development life-cycle, Develop concept, model and process	[62]
Negotiation agents compete for multiple resources	Design agents. Implement agents. Experimental validation	[64]
Methodology for engineering of adaptive multi-agent systems	Define/characterize environment of system. Analyze technology and identify agents	[63]
Use agents to control sensor/robotic systems	Integrate agent capabilities with MAS and physical systems	[65]
Automate the development of an AIS	Implement AIS, automate the process and reconfigure processing algorithms	[69]

Continued on next page

Table 2.5: *(Continued) Characteristics of Multi-agent Systems*

Research Outcome	Research Methods	Author
Traffic simulation with MAS	Integrate environmental info with MAS. Implement traffic simulation	[61]
MAS path-planning algorithm	Formalize MAS path-planning algorithm. Prove low polynomial complexity in time, space and quality	[71]
MAS to exchange info	Implement Dijkstra. Exchange dynamic info between agents	[70]

2.4.2 System-of-Systems

System-of-systems (SoS) is an emerging field: the developer is presented with methodologies that study independent systems with unique attributes, which become complex systems when they work together.

In the work by DeLaurentis and Callaway [72], a lexicon is proposed that divides an SoS into both categories and levels.

> "[A]s good navigating is predicated on the ability to understand and use maps, good decision-making (and problem solving) is predicated on first understanding the problem structure itself and then being able to communicate with others about it."

A formal presentation of categories and levels is presented by DeLaurentis and Callaway and, as modified, appears in Chapter 4, under P_3.

The two primary traits, evolutionary and emergent behavior, present significant insights into the research as it arrives at the current phase and for future development. Drawing on his previous work and using transportation as the illustration

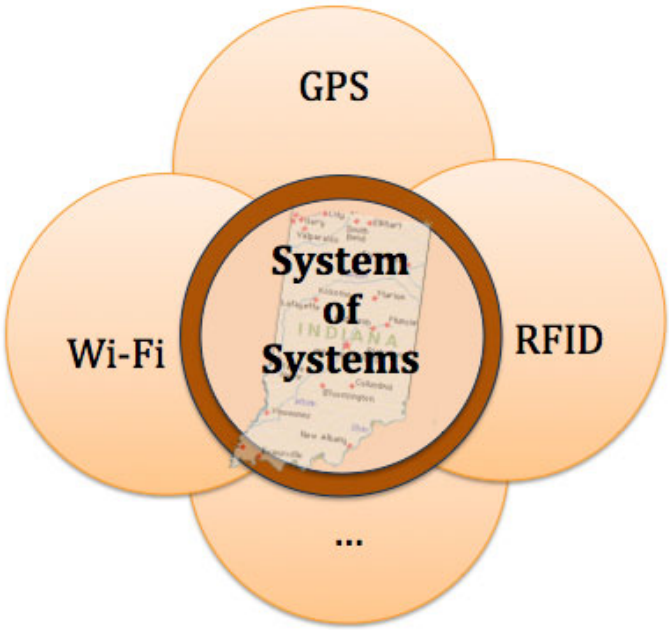


Figure 2.4.: System-of-Systems Concept

topic, DeLaurentis [73] introduced the problems described as system-of-systems and the primary traits noted above.

The problems require articulation, which is the point DeLaurentis made. The code-writing must render path-planning or updating functionalities, considered as intelligent agents, capable of learning and evolving behavior. However, as the code nears the point of delivering this functionality, attention to emergent behavior requires that the user's response to the information be taken into account. System-of-systems accommodates the need for the developer to take each agent's perspective into account, even when moving between field testing and simulation.

Within the model, agents make decisions which affect not only that agent, but potentially many other agents. In Vander Schaaf et al. [74], the agent moved through states "that described its status and logic to direct its transition to new state. It is the user's experience that was the focus of the effort; many of the papers cited depart from this. Thus,

> "within the logical structure, an agent has desires and goals that provide the agents objectives and beliefs, knowledge, and information to determine decisions/actions the agent will make to pursue these objectives [74].

In DeLaurentis and Crossleys work [75] a three-axis taxonomy is proposed to hierarchically categorize and clarify SoS design. The work is proposed to highlight the importance of interactions and dependencies between the systems. This introduces the work of Jamshidi who introduced the notion of system-of-systems engineering (SoSE), which he described as an emerging field with a large vacuum of knowledge [76].

The author offered a list of the most prevalent challenges in dealing with SoS's in the environment: basic definition; theory; management; and implementation. As a brand new field, both authors, (DeLaurentis and Jamshidi), offered their insights into conceptual development more than code-writing.

Jamshidi stated that integration of the systems is the key to a successful SoS, and this integration was achieved through optimal communication and interactions among the systems. The paper also reviewed key issues such as architecture, modeling, simulation, identification, emergence, standards, net-centricity, and control in an attempt to cover open questions of SoS and SoSE.

Modeling an SoS in MatLab requires an attention to system interaction that is qualitatively different from modeling a single agent-based system or a system which offers enhanced navigation to a number of similar agents.

An SoS modeling approach appears in work by Mahulkar et al. [77], illustrated via a Navy Warfighter. To promote "enhanced health and war-fighter performance both afloat and ashore, the SoS envisioned by the Office of Naval Resources seeks to harness "new technologies [to achieve] enhanced war-fighter and system performance with reduced personnel costs as a result of the right information being provided to the right people with the right skills at the right time in the right jobs.

SoS engineering (SoSE) disregards computing power as a scarce resource and focuses on the attention of the warfighter and support personnel (in Jamshidi [76]) as a resource to be marshaled and guarded from distractions. The SoS approach, using

agent-based modeling to simulate involvement of a ship's crew conducting routine functions, is in-line with this research.

The results demonstrated an increase in machine availability due to implementation of intelligent maintenance systems. In the same way, the MatLab programming supports both agent-based modeling to achieve enhanced navigation, with the SoS based restriction that the "right information [must be] being provided to the right people [and computer functionalities] at the right time.

In conclusion, according to DeLaurentis, Crossley and Mane [2] "design is the process of developing a system to achieve a particular goal while managing constraints."

2.4.3 Discussion of Complex Systems

Complex systems in this research are used in the context of multi-agent systems or system-of-systems. The systems are organized and form a higher functioning system when working together as designed. This allows for the possibility of evolving and emergent behavior.

Table 2.6: *Characteristics of System-of-Systems*

Research Outcome	Research Methods	Author
Describe problems of transportation as an SoS	Develop method for transportation SoS. Identify traits and emergent behavior	[73]
Categorization of SoS design	Create taxonomy of SoS. Categorize SoS design	[75]
Develop an SoS engineering methodology	Describe holes in the research. Identify possible solutions. Describe the methodology	[76]

Continued on next page

Table 2.6: *(Continued) Characteristics of System-of-Systems*

Research Outcome	Research Methods	Author
Modeling approach to aspects of a navy war fighter	Address problems in SoS approach. Create ABM for simulation. Analyze results	[77]
Extend computational exploratory model	Improve model. Differentiate and quantify data results. Analyze alternatives	[78]

2.5 Path-Planning

In any navigation application or simulation, the implementation of path-planning determines a route through a physical environment. Typically, algorithms are developed and ranked based on goals and computational complexity. In this research, navigation is achieved by creating a graph to represent a map. The graph is represented as a two-dimensional array of values between one and ten. These values correspond to the cost of moving through the node. Areas that are not traversable are set to infinity. This cost ranking allows the influence of the path-planning results by setting better routes to a lower cost. One of the key features of this research is the assignment of costs of navigation choices.

Ozkil, et al., [79] presented an application that generated and used a hybrid map to perform indoor navigation to enhance robotic movement in a defined and pre-mapped space. The method of the work was two-pronged:

1. map generation and navigation: a local map was generated as an occupancy grid metric map that represented smaller indoor areas. A global map was created as a topographical map using nodes and edges to abstract the areas between the local maps.

2. navigation occurred once the start and end point were known, and was performed using A* algorithm. The path was stored as a sequence of nodes to be followed by the robots.

Generating a custom algorithm for graphing paths from wireless signals supplemented by an acoustic beacon allowed MacMillan et al. [80] to measure informational entropy with movements of the consumer robot, the Roomba. A known trouble area identified was the quality of sensors present in off-the-shelf hardware and the resulting errors due to the high noise levels. Deployment of path-planning algorithms (Dijkstra's algorithm and K-Nearest Neighbor) were not able to overcome these hurdles.

2.5.1 Dijkstra

A 1959 paper written by Edgser Dijkstra identified two problems in connection with graphs. He wrote the paper while associated with the Mathematics Center at Amsterdam, and it supplied an elegant method for finding the single-source shortest path when moving through two-dimensional space. The algorithm for which he is known, functions as a depth-first search algorithm to find the shortest paths from a single starting node to a destination node in a graph [81]. It is explored formally in Chapter 4.

This research has been coded in MatLab and is able to call for a choice of mathematical functions, including both Dijsktra and its next-generation competitor A*, presented in a 1968 publication [82]. The use of competing algorithms in simulating path-planning of a user in two-dimensional space exploits the computing power of MatLab, given the conservation of inputs that the DEVS-based specifications for transitional states guarantees the programmer.

In the work by Yuksel and Sezgin [83], three path-planning algorithms (including Dijkstra) are implemented and compared. The results showed similar computation time and complexity between A* and Dijkstra. The A* algorithm was faster and shorter, but relied more on the input heuristic. When the cost of deviation was

relatively modest, A* was an acceptable choice. In a study by Rampant [84], three algorithms were compared: Dijkstra, A* and Ant Colony Optimization. The authors concluded that both Dijkstra and A* algorithms were efficient, but the author believed Dijkstra's performance would suffer on very large or dense graphs. However, it was still a good choice, as A* was susceptible to the heuristic selected; reliance on user input in this regard affected both error and delay in computation.

In practical terms, the user wants to know both the fastest and the shortest route, in an unfamiliar urban environment. The research examines the user's potential insufficient initial knowledge to secure the promised benefit of A* over Dijkstra.

The paper presented by Wu et al. [85] described path-planning and algorithms for use in indoor navigation for the blind and visually impaired in unfamiliar indoor environments. The path-planning algorithm used an *intelligent map* and a relational data structure called a *Cactus Tree*, combined with a path-planning algorithm (both A* and Dijkstra were used). The approach was to follow paths of convergence and reduce the required accuracy of the underlying positioning and tracking system. The results showed that the Dijkstra algorithm was suitable for path-planning in a navigational application, just as the current research has concluded.

Dijkstra's algorithm is still employed, which attests to its utility: an autonomous system was created by Zhou and Lin [86] which allowed a mobile robot to localize, path-plan and avoid obstacles during movement. The Dijkstra algorithm was implemented for path-planning. The results were validated in both simulation and real world experiments, without, however, the investigator disciplining results via deployment of two competing algorithms.

On the other hand, exploitation of the concept underlying Edgser Dijkstra's work is exemplified in Sun and Li [87], who generated a map using a graph-based grid model over a floor-plan. This allowed the determination of immovable objects (such as walls) and the ability to assign a metric for cost routing. Secondly, navigation was performed using Dijkstra's search algorithm.

The notion of assigning costs to navigation is a fundamental premise of all navigation, although it is not always discussed in the literature as a stand-alone dimension for evaluating results. Humans are complex, and their goal in navigation is not aways the fastest route. The challenge for developers is to enhance, not enforce navigation. Users traveling in a dense urban environment may deviate from the intended goal for any variety of reasons.

2.5.2 Nearest Neighbor

Nearest Neighbor (NN) is conceived as the equivalent of 'interpolation of missing values in one-space.' Belur Dasarathy [88] developed NN as a computer challenge: given a collection of data points and a query point, find the data point closest to the query point (i.e. it's nearest neighbor).

This is particularly useful in mapping or graphing applications needing to determine the nearest point given the current position. The literature has given rise to the nearest-neighbor search algorithm (NNS), a greedy algorithm for finding the closest data point. A more formal treatment will be provided in a later section. For present purposes, the paper by Almeida and Gűting [89] proposed a storage scheme to support the Dijkstra algorithm with k-Nearest Neighbor supplementing information. The results were analyzed quantitatively to show that the proposed data structure is more efficient than previous choices. Beyer et al. [90] implemented the Nearest Neighbor algorithm on various data sets to illustrate that as dimensionality increases, this algorithm is often not the most efficient. This is another instance of mathematics intruding into the opportunities presented by coding and simulating.

2.5.3 Discussion on Path-Planning

Often, declarations of new solutions, are simply improvements to ideas previously advanced. The Dijkstra algorithm, for example, has not been replaced outright, although tinkering has improved it as an operational tool. Advances by the various

communities who exploit this algorithm, are often limited to the increase in computation and run-time in a single application, or overcoming unknowns in variables or challenges. For this reason the review has been limited to research on Dijkstra or Nearest Neighbor algorithms, as this is pertinent to the project.

Table 2.7: *Table summarizing path-planning*

Research Outcome	Research Methods	Author
Hybrid map for indoor navigation, local occupancy grid and topo global map	Map Generation: metric map, annotation, topo map	[79]
Indoor navigation for the blind	Path-planning-Dijkstra and A*. Cactus-tree data structure	[85]
Grid graph-based model for route analysis	Map Generation-grid graph overlay floor-plan. Navigation: Dijkstra	[87]
Efficient shortest path processing	Creating efficient data structure. Calculate shortest path with random sampling and graph Voronoi duals	[91]
Comparing single-source shortest path algorithms	Implementation and Measurement of A*, Dijkstra and Breadth-First	[83]
Comparing single-source shortest path algorithms	Implementation and Measurement of A*, Dijkstra, Ant Colony Optimization	[84]
Use present wireless information to create a path for a mobile robot	Generate algorithm for map creation. Implement prototype. Validate results and discuss errors	[80]
Allow autonomous navigation by a mobile robot	Localize the robot, Path-planing using Dijkstra, Avoid obstacles with laser sensor	[86]
Improved Dijkstra algorithm with k-Nearest Neighbor	Implement k-NN and improve Dijkstra, Measure results	[89]

Continued on next page

Table 2.7: *(Continued) Table summarizing path-planning*

Research Outcome	Research Methods	Author
Illustrate when Nearest Neighbor algorithm is inefficient	Measure efficiency of Nearest Neighbor as dimensionality increases	[90]

2.6 Contribution and Summary

The contribution of the research is in the study of the mechanics of an agent-based system of navigation in a complex environment. The details of switching between individual navigation systems and transport modes is examined specifically.

This study is multidisciplinary in nature. To consider, implement and formalize a model and application prototype, a broad overview of relevant literature was reviewed for this research.

There is a steep learning curve in current research that can be seen as a barrier to further advances. The phrase used is 'jack of all trades, master of none'. For example, there are 73 software platforms listed in Wikipedia for agent-based modeling [92]. Of these, there are 37 different language platforms that these are built upon.

Doing a search for 'simulation software' returns pages for simulation software, discrete event simulation software, dynamic systems, etc. From these pages, there are 60 different platforms. This leaves much of the development choice open to subjective criteria such as: what is available; what is familiar; what is required. Although some topics transcend the implementation (Dijkstra's algorithm), others are specific to a language (e.g. Matlab is specific to numerical computations, the name stands for Matrix Laboratory). It would be difficult to perform some of the specific functions in Matlab in a different environment. Often, papers will not draw from concepts or environmental techniques outside of the development platform. Exceptions indicate that borrowings are restricted to the conceptual level.

These barriers are especially evident when the mathematical foundations of a new algorithm or extension of an existing algorithm is deployed. The authors' presentation does not direct the reader to how the formula will advance the reader's software, but rather the concept generically. Platform implementation is limited to the authors' choice.

No paper presented any translation of technical advances between the major environments or technologies. If intrigued by an application in Java for modeling agents in a given real-time environment, the challenge is to work through a different environment to replicate, if possible, the advance presented in the author's choice of software.

While long accepted with hardware that the machinery in use dictates the user's options, it is a new development in the field of agent-based modeling and application development. The barriers render even literature review problematic. There is no way for an application developer to read thousands, let alone, hundreds of thousands of papers that might yield useful queries. With every author eager to demonstrate that the client's needs have been addressed with some advance in hardware and/or software, identifying *un*fruitful labor or difficulties requiring further research are subordinated by the author's desire for self-promotion. In short, the literature suggests a multiplicity of success-stories and only implicitly identifies basic research required into unknown areas. A summary of the category of each paper is included in Table 2.8.

A literature review should be able to identify specific trends in a problem area or application. Real-time solutions require a deep understanding of the subject and cooperation between researchers. The fields are simply too broad and cover too many interleaved ideas, to have a simple generic list. The solution is often predicated on existing knowledge of the researcher and shaped by personal experiences and familiarity.

Table 2.8: *Summary of Paper Categories*

Localization & Navigation	[56] [57] [54] [13] [51] [53] [93]
... RFID	[94] [12] [13] [16] [20] [22] [23] [25] [24] [30] [11] [19] [95] [96] [18] [97]
... GPS	[28]
... Accelerometer	[27] [28] [29] [32] [33] [34] [98] [99] [34] [100]
... Wifi	[11] [80]
... Other	[12] [19] [20] [21] [25] [30] [31] [34]
... Obstacle avoidance	[21] [54] [80]
Path-planning	[70] [101] [56] [71] [61] [53] [13] [85] [102] [103] [104] [87] [91] [86] [89] [90] [105] [106] [107] [108]
... Nearest Neighbor	[16] [89] [90]
... Dijkstra	[22] [13] [101] [70] [85] [87] [91] [86] [89] [90]
... Other	[56] [71] [85] [83] [84]
... Map generation	[16] [22] [23] [25] [26] [51] [53] [79] [87] [80]
... Methodology	[62] [68] [63] [109] [110] [111]
Algorithm	[11] [12] [13] [21] [24] [30] [31] [83] [84] [80]
Mobile Phone	[28] [29] [32] [10] [4] [5] [112] [6]
Simulation	[61]
... Navigation	[54] [13] [51] [53]
... Formalism	[51] [42]
... Graph Theory	[51] [53] [87] [91] [79]
... DEVS	[51] [52] [42]
... ABM	[54] [51] [55] [52] [13] [43] [45] [42] [47] [40] [113] [114] [115] [116] [74]
... Agents	[64] [57] [117] [118] [65] [66] [70] [68] [56] [69] [61]
SoS	[72] [75] [119] [120] [2] [76] [121] [77] [122] [123] [78] [30] [124] [125]

CHAPTER 3. PATH-PLANNING AND NAVIGATION

This chapter details experiments for each logical phase of the research, from conceptual development through experimental phases. The first section discusses the investigation effort using a consumers hand-held device evaluated in the field, the Samsung Nexus S. The next section involved testing the embedded sensors on the iPhone and finally, the modeling of enhanced navigation through a MatLab simulation. Field-testing via proof-of-concept preceded the simulations as sensors and functionality were explored.

As mobile devices become universal, increased functionality on the devices ensues. Leveraging the characteristics of a smart-phone for new research, allows greater access of the development to more users and limits the cost of entry for any results. This research explores path-planning and navigation on both real devices and through simulation.

During this research a sequence of models and prototypes were implemented (designated P_I - P_{III}). The prototypes denote the individual systems developed (e.g. the Nexus S with AOS, the iPhone 4 with iOS, and the MatLab models). The progression of complexity and ideas is presented in this chapter.

P_I was a proof-of-concept to study path-planning and navigation as well as the embedded sensors on a smart-phone. P_I was developed on the Samsung Nexus S / Android operating system for several reasons:

1. the Android operating system contained a built-in NFC compliant RFID reader and writer;

2. accessibility of geo-aware overlays using Google Earth;

3. the Android SDK contained functionality for Google maps, which allowed access to existing maps and path-planning algorithms; and

4. the development environment is in Java, which is widely-used and more open than the iPhone.

P_2, the second prototype, was a combination of simulation and mobile sensor exploration. Localization using the embedded sensors of the Apple iPhone was examined, as well as simulation of a multi-agent system for path-planning on a phone. The iPhone was selected for this experiment as it contained both an embedded three-axis accelerometer and three-axis gyroscope. The API contained appropriate functionality and filtering to gather data from these sensors to study if the data would be useful. Although the result of the embedded sensors on the iPhone was not a usable solution (due to reasons discussed in later sections), it was still a useful exercise. It allowed experimentation with the iPhone and its environment. Additionally, the first agent-based model simulation was designed and successfully deployed based on i-Phone acquired data.

Finally, the third prototype (P_{III}) was a progressive simulation involving localization, path-planning and navigation in both indoor and outdoor models. The prototype was not developed on a mobile device, as the simulation environment allowed much greater range and freedom to explore variable relations and influences. It also limited the noise involved in the system which could overwhelm the data.

To recap: two physical and one virtual prototype were deployed; the physical prototypes involved the Samsung Nexus S and a Java-based application; other results were obtained from the Apple i-Phone running iOS. “Proof-of-concept is used interchangeably with prototype in this presentation.

3.1 P_I: Path-planning on the Samsung Nexus S

The first prototype was implemented on the Samsung Nexus-S phone. The system coded in this research was based in Java and implemented the Android and Google Maps frameworks. This prototype generated a system-of-systems that allowed a user to transition autonomously between individual navigation sensors as appropri-

ate. An example could be walking across campus using GPS and wireless systems and then transitioning to Wifi and RFID system indoors.

Motivation: One of the overarching goals of the research was to investigate a system of multiple navigation sensors to enhance urban navigation. To do this, the strengths and weaknesses of various mobile platforms and embedded sensors was explored. By field testing different combinations of hardware and software, the costs and benefits pertinent to a solution involving a fusion of sensors would be clearer. The system was coded in Java and implemented the Android and Google Maps frameworks. The motivation of this prototype was to generate a system-of-systems (SoS) that allows a user to transition autonomously between individual navigation sensors as appropriate. An example could be walking across the Purdue campus using GPS and Wifi systems and then transitioning to Wifi and RFID systems indoors, where GPS is often unreliable or unavailable. There are several benefits to this data fusion approach:

- the ability to localize more precisely depending on location requirements;
- the ability to move autonomously through systems without user intervention; and
- the ability to add new technology as available to improve existing system-of-systems.

3.1.1 Methodology

The methodology used in this experiment was localization and path-planning done via the Google Map SDK. The user's position was marked on the Google map based on either GPS or Wifi. If an RFID tag was detected, the related floor-plan was appropriately sized and shown over the location on the Google Map. The following steps were taken:

1. Get the latitude and longitude of each buildings image for correct placement as the overlay;

2. Write the correct latitude and longitude of each RFID tags placement;

3. Software will be developed for the Nexus S phone.

   Locate current location of user;

   Read the RFID location;

   Calculate the size of the building;

   Overlay the building on Google Map;

4. Update current location of user based on RFID tag.

3.1.2 Design

The system was designed with several factors in mind: ease of use; inclusion of embedded sensors; limited external hardware or infrastructure requirements; and a software development kit with application programming interfaces for implementation.

Utilizing the features of the Nexus S smart-phone, both hardware and software were useful to the application. The Android software has an API for Google Maps, as well as the ability to localize using both GPS and Wifi. The phone hardware contains built-in NFC compliant RFID reading and writing support. This allowed the addition of passive RFID sensors to the model.

Although much of the map functionality is provided through the API, there are a considerable number of constraints when localizing, e.g.:

1. Signal strength: Wifi varies in any physical environment and Purdue's signal strength is no exception. It is better inside some buildings than others, better on some floors than others, and can drop off in the space between building

leaving holes of coverage which will be addressed in the simulation phase in a later section.

2. Elevation: A known constraint of GPS and Wifi is the difficulty in determining the vertical dimension of a location. Latitude and longitude are the same no matter how high or how low one is in that position.

Wifi can address this limitation by field testing and mapping signal strengths and overlaps among access points. Signals are subject to interference and in a dynamic environment, this can be problematic. This is a broad and open research question, and is outside the scope of this experiment.

3.1.3 Implementation

The Samsung Nexus S comes with a variety of embedded sensors and on-board chips. Those relevant to this research are a high performance, single GPS chip (ST Ericsson GNS7560), an NFC compliant RFID controller (NXP's PN544 NFC chip), and a low-power Wifi chip (Broadcomm BCM4329GKUBG Wifi). Although GPS and Wifi are common capabilities among smart-phones, an RFID tag reader and writer is novel. For this research a passive tag (UPM Raflactac) is used.

The experimentation on the Nexus phone was a combination of localization using GPS and Wifi, and the addition of passive RFID tags. In this experiment, the elevation and building information was obtained form the addition of NFC compliant passive RFID tags. NFC is a subset of passive RFID tags which uses inductive-coupling (i.e., loosely coupled inductive circuits share power and data over a distance of a few centimeters) [126]. NFC compliance indicates the implementation of an ISO accepted standard [127]. The tags work in the 13.56 MHz High Frequency (HF) range, which allows the use of coils constructed with printed ink and an EEPROM. This lowers the cost of individual tags and supports the goal of using less or inexpensive additional equipment. This experiment uses the NFC Data Exchange Format (NDEF)

specification, a common data format which supports four record types; text, URI, smart poster and generic control.

Within this experiment, the physical environment had no permanent RFID tags installed. The tags were programmed, temporarily placed, and tested as explained below.

The model was implemented in Java through the Eclipse IDE. Development began using a freely downloadable Android software development kit (SDK). Google also provides an application programming interface (API) for their mapping software functionality. This allows Google map, a universally familiar format, to localize a user through GPS or Wifi. Sensitivity parameters and provider information are configurable within the code.

Once implementation of localization with Wifi and GPS was working, the portion involving RFID was studied. The ability of the embedded sensor on the Nexus to detect incoming RFID signals was utilized in the programming. This is implemented as a Java event, which allows the system to detect ambient change in state. In this case, an 'event' occurs when the system detects the reading of an NFC compliant tag. When it does, the software reads the text field information in, which for this research stored the building and room number within the 64 kilobytes of memory available to the tag.

Additionally, with the ability to determine elevation, a need to see more than the gray outline of the building as provided by Google maps is necessary. The Purdue College of Technology website has a facilities page with downloadable JPEG's of the Maurice G. Knoy Hall of Technology floor-plans. Overlaying the image on the Google map allows the navigator to obtain a location inside a building.

The floor-plan obtained from the website comes in the JPEG format as 14 inches by 8.5 inches. While this provides clear imagery for the website, it is exceedingly large for a smart-phone display. To have functionality, the image must resize as the user zooms in or out. Additionally, there is a point at which the zoom is too small for the floor-plan to provide any useful information. A collaboration occurred to solve

the zoom issue described as follows: in addition to the correct geographic location of the the floor-plan, the image must also be appropriately sized based on the current size of the map. To achieve this, the position on the screen of two points with corresponding coordinates to the image's top left and bottom right geo-referenced corners was obtained. While the screen position is required for sizing the image correctly, it also provides reference to whether the building is currently visible in the map. If not, the image does not need resizing.

To determine the correct size for the image, a comparison (in Pixels) must be done between the actual image size and the size of it on the screen. To compare these values, the diagonal measure of the image must be done both for the actual image and the place in the map of the image. Equations are shown in Eq: 3.1 and 3.2. As this prototype was deployed on a mobile phone, the threshold value of 0.1 is selected as the lower bound of the zoom. Anything smaller would have no purpose and lack details.

For two points $p_1[x_1 and Y_1]$ and $p_2[x_2 and Y_2]$ the distance between them is defined by the function:

$$distance(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

The zoom function is then given by:

$$zoom = \frac{distance(topLeft, bottomRight)}{\sqrt{imageWidth^2 + imageHeight^2}} \tag{3.2}$$

The College of Technology is located in the Maurice G. Knoy Hall of Technology and is shown in Fig. 3.3. The building is shown as a satellite image with the floor-plan overlay. The improvement in this map is the ability to recognize the floor based on RFID tag. In the experiment, the RFID tags were placed at the door to the stairwell of the first and second floor. As a user navigated through the building and moved up or down the stairs, a scan of an RFID tag would allow a JPEG floor-plan overlay to switch to the appropriate floor. This allows easier navigation not only indoors but on multiple floors and elevations as well.

The floor-plan must be placed on the Google map in the right location, known as geo-referencing. When laying a rectangular image over the map, two locations must be known, opposing corners of the image. To determine the location, the JPEG image was used in Google Earth. By sizing the image appropriately to the satellite image on Google Earth, each corner of the figure's latitude and longitude could be obtained to allow for the proper placement. Fig: 3.2 shows the overlay in Google Earth of the Richard & Patricia Lawson Computer Science building in the correct location. The Java code took the known upper left and lower right corner and placed the image between them. The default was to set the first floor image as current. The addition of the room number allowed the application to differentiate between floors in buildings that are more than a single story (100's indicate the first floor, 200's indicate the second floor, etc.).

Figure 3.1.: LWSN floor-plan overlay

If a different floor was discovered, the current floor-plan overlay was replaced with the appropriate second, third or fourth floor image set to the appropriate size.

Figure 3.4 shows a map of the Purdue University Campus. Floor-plan image overlays are included for Lawson Computer Science and Knoy Hall of Technology. The map shows a Google-generated walking path (the blue line) from Lawson via defined road to Knoy.

Figure 3.2.: LWSN floor-plan overlay

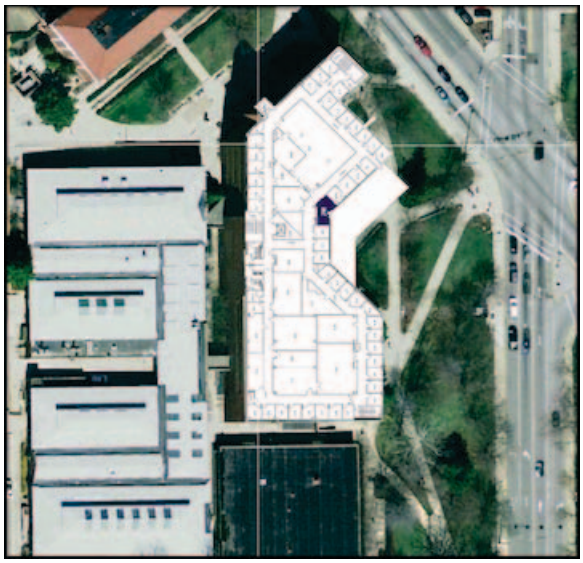


Figure 3.3.: KNOY floorplan overlay

In addition to performing sensor localization, the system was designed to record and display both the current user location and the provider in use. The image in Fig:3.4 shows the path walked by the user from Lawson Computer Science to Knoy Hall of Technology. In the actualized path, the colors changed in response to the provider in use: red for Wifi; yellow for GPS; and purple for RFID. Additionally, the path was not limited to only defined roads.

Figure 3.4.: The path on Purdue Campus between Lawson Computer Science and KNOY

3.1.4 Results

The proposed proof of concept provided a data fusion of existing techniques and new technologies to provide an improved navigation application. The system was implemented on existing and publicly available devices. This limited the need for additional or costly equipment. By using both existing devices and ISO standardized equipment, the system is open-source and easily modifiable. The system was able to track the user across campus and within the building as well as provide the current floor-plan based on the user's location.

3.2 P_{II}:Path-planning on the Apple iPhone

The second prototype was created using the fourth generation iPhone and its internal sensors. The fourth generation iPhone was one of the first smart-phones to include an essential sensor for inertial navigation: a three-axis gyroscope. This sensor, in addition to the three-axis accelerometer that the previous generations contained, made it a good candidate to study in an inertial navigation system. Using a general public-oriented device was challenging from a sensor standpoint because of the low

quality of the embedded sensors. Additionally, the complexity of human movement increased the difficulty of research in this area.

The benefit of the iPhone was that it was widespread and relatively low cost. This narrowed the cost gap between any research results and the end-user. Additionally, a multi-agent simulation was explored on the iPhone for path-planning. The motivation of the simulation was to explore the appropriateness of an agent-based model for mobile path-planning applications. Each agent in the system was responsible for a set of specialized tasks. This allowed the implementation to be easily modified to individual needs for both ease and usability. Implementing the model as a multi-agent framework, enhances adaptation to different goals or scenarios.

3.2.1 Methodology

For the experiment, two processes were used:

- (1) use of the fourth generation Apple iPhone, with the computation and recording information done locally; and
- (2) computation and logging off-loaded from the iPhone via UDP packets over the local Wifi connection to a computer running LabView.

The relative projection was calculated first. To obtain useful information about the phone's movement, it must be expressed relative to the earth's referential. To do this a measure of the acceleration in the earths' referential was needed. The gyroscopes of the iPhone measured the attitude of the phone. Knowing the attitude, the acceleration vectors (expressed in the referential of the phone) could be projected (to the referential of the earth).

The theory behind inertial navigation systems is well known: accelerometers and gyroscopes are used to measure acceleration relative to the earth. Using the approximation that this referential was Galilean, two integration steps were used to get the position from the acceleration.

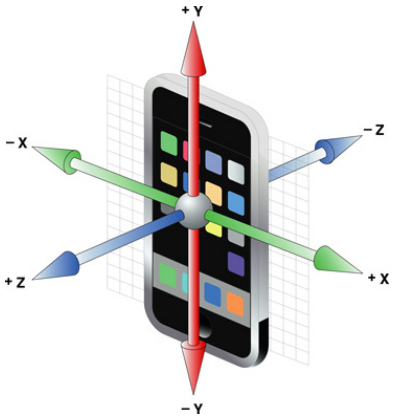


Figure 3.5.: This figure shows the coordinate system attached to the iPhone in which the accelerations are expressed

(1) Integrate the acceleration to obtain the speed:

$$\int_0^t a_x(t)\,\mathrm{d}t = v_x(t) \quad v_x(0) \tag{3.3}$$

$$\int_0^t a_y(t)\,\mathrm{d}t = v_y(t) \quad v_y(0) \tag{3.4}$$

$$\int_0^t a_z(t)\,\mathrm{d}t = v_z(t) \quad v_z(0) \tag{3.5}$$

(2) Integrate the speed to calculate the position:

$$\int_0^t v_x(t)\mathrm{d}t = x(t) \quad x(0) \tag{3.6}$$

$$\int_0^t v_y(t)\mathrm{d}t = y(t) \quad y(0) \tag{3.7}$$

$$\int_0^t v_z(t)\mathrm{d}t = z(t) \quad z(0) \tag{3.8}$$

In addition to the theory being well known, the issue of double integration causing a squared error is also well known. Using sensors always includes noise.

Despite filtering, an error due to this noise is present. Integrating a single measure will result in a linear drift after the first integration as seen in Eq: (3.10) and a squared drift after the second shown in Eq: (3.11).

Definition 3.2.1 *Let A_x be the value of acceleration measured along the x-axis, a_x the real value of the acceleration and $\epsilon\psi$the error:*

$$A_x(t) = a_x(t) + \epsilon\psi \tag{3.9}$$

$$\int A_x(t)\mathrm{d}t = \int (a_x(t) + \epsilon)\mathrm{d}t = v_x(t) + \epsilon t + \alpha\psi \tag{3.10}$$

Definition 3.2.2 *To simplify the equation, assign the integration constant $\alpha\psi = 0$, observe the linear drift ϵt due to the error in the measure of the acceleration. This time a squared drift ϵt^2 is observed.*

$$\iint A_x(t)\mathrm{d}t = \int (v_x(t) + \epsilon t)\mathrm{d}t = x(t) + \frac{1}{2}\epsilon t^2 + \beta\psi \tag{3.11}$$

Additionally, a second component implemented was a multi-agent simulation of path-planning on the iPhone.

3.2.2 Design

An overview of the system used is shown in Fig: 3.6. Steps of the system are also detailed.

The iPhone has two embedded sensors that were utilized, an accelerometer and a gyroscope. The accelerometer was an ultra-low power digital three-axis model by STMicroelectronics (LIS331DLH). According to the data-sheet of the constructor the accelerometer can be configured to measure acceleration data between $\pm 2g/\pm 4g/\pm 8g$. The experiments shown in Table 3.1 indicate that Apple configured it in the $\pm 2g$ range.

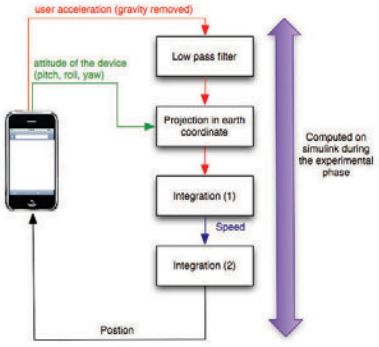


Figure 3.6.: Overview of the system

Table 3.1: *Sensor characteristics*

Accelerometer		Gyroscope	
Range (g)	± 2	Range (dps)	± 250
Acceleration noise density ($\mu g/\sqrt{Hz}$)	218	Acceleration noise density (dps)	0.03
Bandwidth (Hz)	25	Bandwidth (Hz)	40

A three-axis gyroscope by STMicroelectronics (ST) was included in the iPhone 4 (L3G4200D). A gyroscope is used to determine the rate an object rotates. Integrating this data over time measures the pitch, yaw and roll of the phone - again due to integration in this step, the measure suffers from drift.

According to the data-sheet of the constructor the accelerometer can be configured to measure acceleration data between $\pm 250dps/\pm 500dps/\pm 2000dps$. The experiments shown in Table 3.1 indicate that Apple configure it in the $\pm 250dps$ range.

The drift observed was almost nonexistent for the pitch and roll when the iPhone was standing still. The API used to access the roll and pitch of the iPhone implemented an efficient algorithm to remove the drift using the direction of gravity (obtained by the accelerometer). The drift is more important for the yaw, but in the

experiment the drift observed was less than one degree over ten minutes when the iPhone is stationary.

Drawing a functional Inertial Measurement Unit (IMU) from the sensors was a challenge. Although the sensors embedded in the iPhone 4 were more accurate than the one in the previous versions, they still delivered noisy results.

In iOS version 4, Apple introduced a new framework called CoreMotion [128]. This framework allowed access to low level information (i.e. the rotation rates of the gyroscopes or the raw acceleration data). It also permitted access to higher level data (i.e. the user acceleration with gravity filtered out or the pitch, roll and yaw).

The iPhone provided a measure of acceleration where the constant due to gravity had been removed. The acceleration due to the motion was a low frequency signal, so the signal was isolated from the noise using a low pass filter.

To determine the projection, the following matrix was used. It is the inverse of a rotation matrix with the three Euler angles of the phone ($= yaw$, $\theta\psi = pitch$, $\phi = roll$).

$$\begin{bmatrix} c\theta c & c\phi s & + s\phi s\theta c & s\phi s & + c\phi s\theta c \\ c\theta s & c\phi c & + s\phi s\theta s & s\phi c & + c\phi s\theta s \\ s\theta s\phi c & & \theta c\phi c\theta\psi & & \end{bmatrix}$$

Figure 3.7.: Projection matrix, cosines abbreviated with c and sinus with s

Integration (1) : The first integration indicated the use of the Zero Velocity Update principle to remove or reduce the drift.

Integration (2) : The second integration provided pertinent information regarding the position of the user. This was tied to the path-planning portion of the experiment, and results shown for this are broad and general.

An initial calibration was required:

- To compute angles of rotation from the rotation rate provided by the gyroscopes, the origin of the angles must be defined.

- The pitch and roll of the iPhone are zero when parallel to the ground and face up. An initial calibration was required. The accelerometer and a high pass filter were used. This provided the direction of gravity, expressed in the referential of the iPhone. Basic trigonometric functions were used to get the corresponding pitch and roll of the iPhone shown in the equations below.

- Initial speed: the constant in equation (3.3) corresponded to an integration constant.

- Initial position: the constant in equation (3.6) corresponds to an integration constant. To make the results useful, a known initial starting position was necessary. This portion was offloaded to the path-planning and navigation portion.

The multi-agent simulation architecture was designed with other considerations. An overview of the system (Fig. 3.8) details which agents were used and updated the data held in the blackboard and the messages they were able to send.

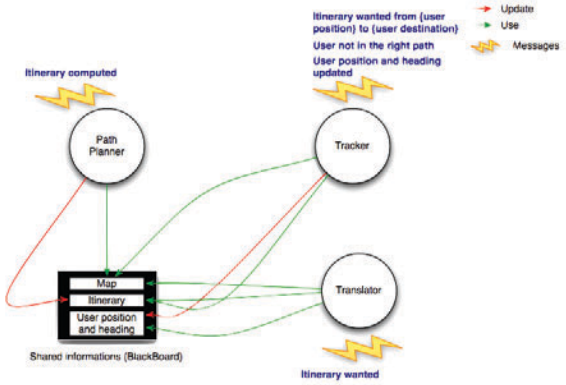


Figure 3.8.: Overview of the agent-based model

The *path-planning agent* generated the itineraries based on departure and destination points on the map, the *translator agent* generated the interface between the system and the user and the *tracking agent* displayed the absolute position. The information was corrected according to mapping information (i.e. in car navigation if the absolute position was not shown on a road we would correct it). The *tracking agent* tracks if the user was following the computed itinerary. If the user deviated from the path, the agent issued a message to trigger the recomputing of the itinerary.

The shared information cache (e.g. the blackboard) was storage space where agents put information needed by other agents in the system. Each agent was able to read and update information. There were several pertinent pieces of information stored. A map was used by the *path-planning agent* to compute the itinerary, the *tracking agent* to correct the position if needed and the *translator agent* to provide contextual map information to the user. Next an itinerary was created and updated by the *path-planning agent*. It was used by the *tracking agent* to correlate the user path to the computed path. It was used by the *translator agent* to provide information about the itinerary if needed. Finally, user position and heading updated by the *tracking agent* was used by the *translator agent* to provide the user with information regarding their position.

The *path-planning agent* has two states: waiting and computing path. In the waiting state, the agent listened to the messages from other agents. When a message "itinerary wanted" was received, the agent moved into the state, computing path. The agent used the departure and destination data provided to compute an itinerary. Once the computation is complete, the agent sent an "itinerary computed" message and updated the corresponding itinerary on the blackboard. The agent then returned to the waiting state.

The *tracking agent* functioned as the clock of the system. When a new position was detected the *tracking agent* was in charge of detecting the change. The *tracking agent* moved between the following states: tracking, mapping, or wrong path. In the tracking state, the agent acquired the sensor data and computed the position of the

user in the appropriate coordinate system. When a new position was computed, the agent returned to the mapping state where the position was adjusted.

In the mapping state, the *tracking agent* compared the position of the user to the itinerary. When the mapping of the user position was complete, the agent sent the message "user position and heading updated." If the user was in the path corresponding to the itinerary, the *tracking agent* returned to the tracking state, otherwise the agent changed to a wrong path state. When the agent entered this state a "user not in the right path" message was sent and the agent retrieved the user destination from the blackboard. The *tracking agent* requested a new itinerary with the message "itinerary wanted" plus the user departure and destination positions. When completed the agent returned to the tracking state.

The *translator agent* initialized to a waiting state. The agent listened to messages from other agents and user inputs. If new information was received, the agent transitioned into an updating interface state. When user input was detected, the *translator agent* moved to a state dependent on the new input. In the get itinerary state, the agent converted the user input into two positions on the map: the departure and the destination. If the departure corresponded to the current user position, no input was required for that measure. When the positions were processed, a message "itinerary wanted", plus the selected departure and destination coordinates, was sent. The system returned to a waiting state. The framework of the model allowed transitions to be implemented between the waiting state and updating interface state to fit the specific characteristics of the modeler's needs.

All agents had the same communication capabilities: the ability to post and read textual messages to the shared blackboard; and the ability to send/receive broadcast messages with textual data. The organization was implicitly known to all agents, as they shared the same unambiguous communication protocol. The difference in communication method selected (i.e. post and read vs send and receive) was based on the needs of the data. If the data was central to the model with an important lifetime it was posted or read. However, if the data was transient, then it was broadcast.

Each agent had individualized functions. The *path-planning agent* used the Dijkstra Algorithm [129] to find the shortest path between departure and arrival nodes selected by the user. In this first model, the positioning data needed was simulated. The path followed by the user was defined when the application started (the path may or may not correspond to the itinerary) and the application changed the virtual position of the user by time-step. The *translator agent* functioned as a Human Computer Interface (HCI). In the model the end user (a human) was presented with visual information on the screen about the map and itinerary.

3.2.3 Implementation

Only data provided by the embedded sensors of the iPhone were used in this prototype. The initial experiment kept all measuring and filtering on the iPhone.

For greater flexibility and easier future change, this intial setup was modified. Filtering parameters and data was sent via UDP packets over the Wifi network to a computer running LabView. LabView provided processing functionality to determine the most appropriate parameters for filtering and integration.

Attachment of the device to the body: The body has many degrees of freedom (DOF). This presented a challenge for the system, as high DOF mean unwanted movement in the measure (e.g. trembling or vibrations). Initially, the iPhone was attached to the foot, a body part with a consistent and stable motion when walking, which allowed the Zero Velocity Update principle to remove the drift due to the integration of the speed. The basis of this principle was that as a person walks, one foot moves and then stands still. If the end of a step and the beginning of the next step can be detected, the speed of the foot during the period is zero. This allowed the speed to be re-calibrated, decreasing the effect of the drift.

A solution considered to detect a step :

- Detect the minimum of the acceleration norm:

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.12}$$

- Detect the minimum of the gyroscope rotation rate.

Re-initialization to account for drift

---

begin

  if $|a|$ *is* $<$ *threshold* **then**

    reset Initialization variable to true

Although having the phone strapped to the foot was not a practical solution, the idea was to identify and categorize movement. If the iPhone was in the hand or pocket and movement was categorized, there would still be a moment when a method (such as the Zero Velocity Update) would be effective. As the movement of the body was consistently inconsistent, this was a difficult challenge.

For the simulation portion, the agents were implemented in C and Objective C and embedded on the iPhone. The Cocoa Touch framework provided by Apple was also used.

The map and itinerary were formalized as a directed graph for single-source shortest paths. In this model a textual representation of a list of nodes and a list of edges was used, plus the weight of each edge to define the graph. The graph was defi by the equation: $G = (V, E, w)$, where V is a set of vertices (or nodes) and $E \subseteq VxV$ is a set of edges. As E contains ordered pairs, the graph is directed. By adding the variable (w), weighting of the edges was implemented. Variable (P), is defined as the path between vertices. If P consists of edges $e_0, e_1, ..., e_{k-1}$, then the length of P, denoted $w(P)$ is calculated using equation 3.13:

$$w(P) = \sum_{i=0}^{k-1} w(e_i) \tag{3.13}$$

The blackboard was the storage cache accessible by all the agents. This was implemented using the SharedInstance/Singleton pattern. This ensures that only one instance of the blackboard existed at run-time. The Agent Class contained the common behavior of all agents in this model. To enable inter-agent communication the

agents used an instance of the MessagingProxy class. The behavior of the MessagingProxy is described below.

Each agent was subdivided into classes: Translator; Tracker; and PathPlanner. The classes defined the general actions the agents were required to implement; the states the agents could be in; and the binding between the messages and the functions. As this was modeled in Objective-C, none of the classes were defined as abstract. The following classes implemented the actions of their superclass: MyTranslator; MyTracker; and MyPathPlanner. By using this model, a level of abstraction was created that allowed easy modification of the implementation details, depending on the context of the application.

The MessagingProxy class was the messaging interface of each agent. This component implemented the communication capabilities of the agents: the ability to broadcast messages using the communication channels available. In the CocoaTouch Framework the broadcast mechanism is called NSNotification. NSNotification broadcasted an index (the integer identifier) and a dictionary containing any number of objects. To transform the message (and data) that any object wanted to send, it was placed into a dictionary as an NSNotification object. This object was dispatched by the NSNotificationCenter.

To receive a message, the process was reversed. The MessagingProxy listened to the NSNotifications broadcast using NSNotificationCenter. The proxy breaks down the NSNotification object received into a message. The delegate pattern allowed the agents to be informed by their MessagingProxy when a new message has been received. This messaging architecture made it easy to implement other communication channels; to do so involved implementing additional communication channels as needed; and the MessagingProxy required implementation details of encoding and decoding a message.

When the application launched, the user was asked to enter both the departure coordinates by touching the nodes on the screen. The *translator agent* (currently in

the get itinerary state) sent the message "itinerary wanted" as well as departure and destination nodes selected. It then returned to the waiting state.

The *path-planning agent* received the "itinerary wanted" message along with the coordinates and moved into the computing path state. It computed the shortest path between the departure and arrival nodes. Once the computation was complete, the *path-planning agent* returned to the waiting state and sent the message "itinerary computed." The *translator agent* received the "itinerary computed" message, and switched to the updating interface state. In this state the itinerary was displayed on the interface as a green line between the departure and arrival nodes.

To simulate the path in this model, the user touched the nodes on the screen in succession to simulate a user moving along an actual route. The path was designated with purple nodes. The *translator agent* returned to the get simulation path state and sent a message "simulation path acquired", plus the route information. The *tracking agent* received the information, stored it and began sending positioning signals corresponding to the simulated data. The states and messages corresponding to this step are not described further in this model as they are artifacts of the simulation.

When the virtual user moved one time-step, it corresponded to a change in user position, and was logged as a normal step on the computed itinerary. The *tracking agent* in the tracking state generated a new position and heading for the user. As the user had taken a normal step, the *tracking agent* sent a "user position and heading updated" message and updated the user's current position and heading on the shared blackboard. The *translator agent* received the message and moved into the updating interface state. The new position of the user, the direction to turn at the next intersection and the remaining distance are updated and the display refreshed.

If the virtual user in the time-step changed position and stepped out of the itinerary, it is considered not a normal step. The *tracking agent* and *path-planning agent* maintain the same behavior as the normal step. The *translator agent* received the "user position and heading updated message", however as the user was not on the computed path, the *tracking agent* shifts into the wrong path state. It sent a

"user not in the right path" message. The *tracking agent* received the destination of the itinerary from the blackboard and sent the current position and the message "itinerary wanted" to the *translator agent*. The *translator agent* reacted to this message by changing into the updating interface state and displaying an alert on the screen in red text.

3.2.4 Results

System with filtering embedded on the iPhone: The prototype used only the sensors from the fourth generation iPhone: 3-axis accelerometer and 3-axis gyroscope. The measures taken during several of the test runs are shown in Fig: 3.9 - 3.11.

For the initial run, movement was limited to a straight line with the iPhone strapped tightly to the foot to reduce the amount of noise due to the complexity of human movement. The acceleration along the x and y axes are shown in Fig: 3.9. A simple low-pass filter was used with a cutoff frequency of 2Hz and a trapezoidal algorithm for integration. The Zero Velocity Update method was not implemented yet so the drift for the speed was important.

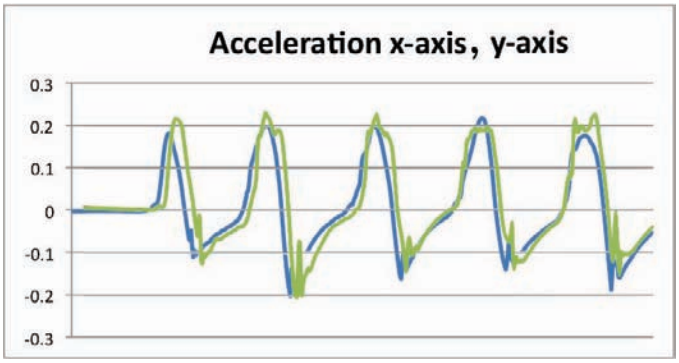


Figure 3.9.: This figure shows the accelerations along the x and y axis

System with the iPhone forwarding data to LabVIEW: In this trial, data from the sensors was sampled at a frequency of 60Hz. The data was sent as a string via UDP packets over Wifi to LabView. The string was made up of acceleration along

the x, y and z axis and pitch, roll, yaw. These measurements were passed into the processing element where they were projected relative to the correct referential. The data was filtered using the Butterworth low-pass filter. To create the effect of the Zero Velocity Update, a re-initialization occurred each time the norm of the acceleration was near zero between the first integration and the second. Data was exported to both a graph and a file. Although these results would not be accurate for long durations, when combined with other information they could potentially be useful to assist in indoor location estimation.

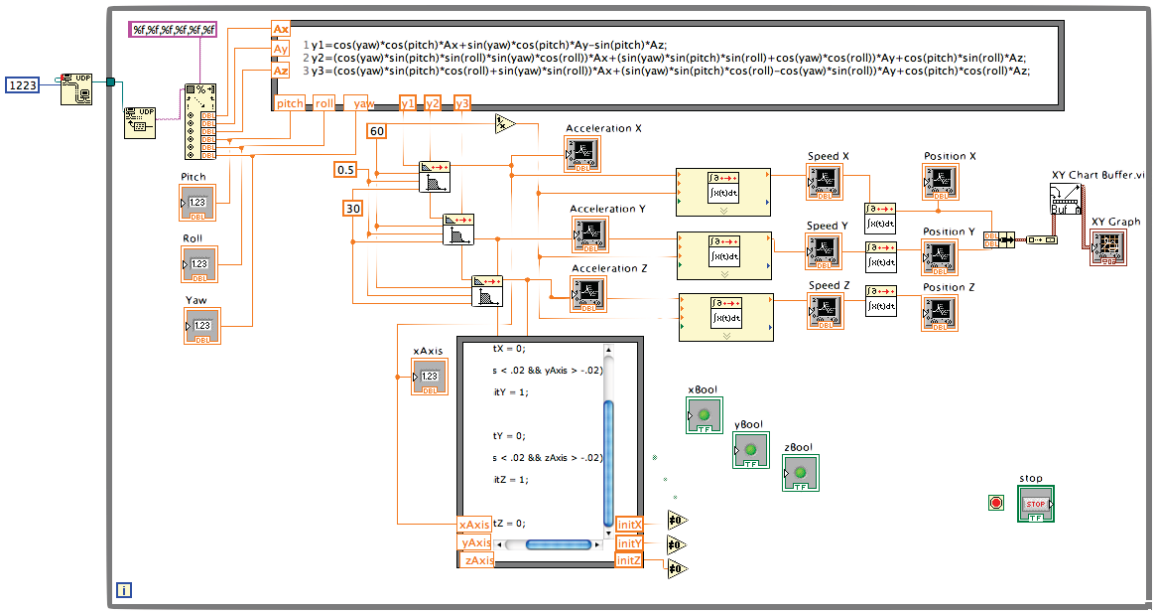


Figure 3.10.: This figure shows the signal processing in LabView

Fig: 3.11 shows an experimental run where the phone was strapped tightly to the foot. A straight line, approximately five steps long was walked, a directional change of 180 degrees occurred, and the same straight line was walked to the origin. Although the position was not projected onto any mapping feature, it still showed the anatomy of movement equal to the actual movement with the expected drift.

Clean results that match what was expected were gained in the shape of the curve, however the values are far from being accurate. To get clean curves, a low-pass

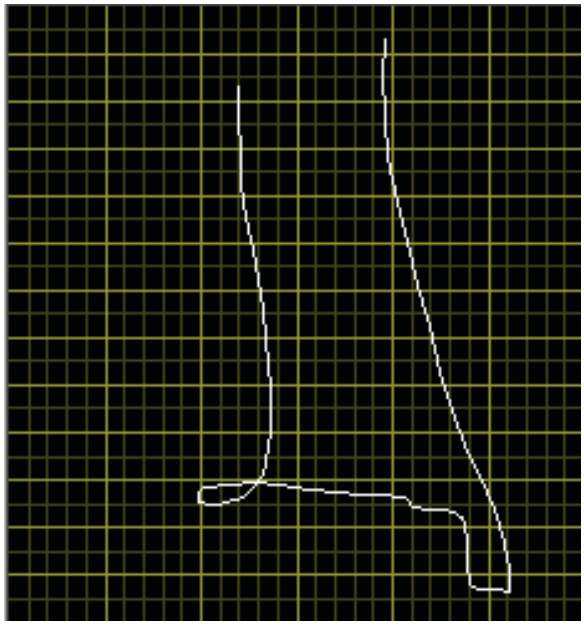


Figure 3.11.: This figure shows a person moving 5 steps forward, turning, and 5 steps back

filter was applied with a very low cutoff frequency. This caused the noise to be very low but also filtered a part of the signal required for accuracy. The filtering would need improvement to get accurate or even useful values.

To improve step detection, the derivative to detect the minimum threshold (of the acceleration norm or the gyroscope's rotation rate) instead of using a simple threshold was studied. Although this would be more costly in term of processing (which is a scarce resource on a smart-phone), it would be much more accurate. This work concluded with the realization that the sensors were not nearly exact enough to get the type of precision required for indoor navigation. Additionally, human locomotion was too complex to correctly categorize movement of a user holding the phone to apply the zero velocity update.

The simulation was a system implemented as a proof-of-concept. As such it was implemented in a simplistic manner to verify functionality. The model was implemented on a ubiquitous smart-phone; however the model could have been developed

from data acquired through other platforms. The system was designed to be easily adaptable to a multitude of environments and situations. By using a multi-agent system, the basic system is solid and highly flexible within the application.

The model was extracted from the development of previous work on an indoor positioning system designed for smart-phones. Future work involves moving from a system designed for a theoretical environment with simple map data and simulated positions to a system using the indoor positioning method we are developing and actual geographical data in the context of a building. To accomplish this will require improvements regarding the way path-planning is implemented so that the model is able to handle larger graphs. One possible solution may be embedding the geographical information (i.e. the predecessor list for certain precomputed paths) within the application. The result would be only small parts of the itinerary requiring real-time computations. Finally, the communication protocol could be improved with better error handling in the system.

3.3 P_{III}: MatLab Implementation & Simulation

For the third prototype, a model and simulation was developed to implement a multi-agent-based complex system for navigation. This model was developed as a simulation for better control over the environment and the noise factors. By controlling the environment, the measurements are presumed to be valid and do not include any external factors.

For this model, navigation in a complex campus environment is viewed as a system-of-systems (SoS) problem. The SoS is made up of independent systems, each able to aid a user in navigation, utilizing whichever system, sensor or tool is available. Using intelligent navigation sensors and techniques (RFID, wireless, GPS, embedded sensors on a mobile device, IMU) based on current availability in combination with an adaptive technique to switch between them brings the possibility of a navigational Multi-agent SoS (MaSoS).

Abstracting the problem provides four distinct areas of the problem to be examined;

1. stakeholder;
2. resource;
3. driver; and
4. disruptor.

Additionally, the interactions between these four areas are examined. This allows for a big-picture view of the problem. In turn, this assists with a better understanding and facilitates the transition from defining the problem to implementing the solution. The abstraction taxonomy was based on the work of DeLaurentis [75].

1. The stakeholders are the people or organizations who have a significant interest in the results of the research. The drivers of a system-of-systems are the activities of the stakeholders that determine the overall demand for the service. For the complete navigation system-of-systems, the stakeholders are anyone who uses a navigation system, while the drivers are the activities of the stakeholder that require the use of navigation. These activities are the drivers and are defined by what the stakeholder wants to do and governs how the stakeholder will behave. For the navigation system-of-systems, they can be social, economic, or political. For example, a social driver could be an everyday person who would like to meet their friend at a new location. An economic driver could be represented by a package delivery service like FedEx or UPS looking for more effective ways to guide their delivery personnel to the package destination. Finally, political drivers would be well represented by law enforcement and defense personnel whose guidance needs could potentially be mission critical.
2. Resources of a system-of-systems are the "supplies used by the stakeholders in the system-of-systems. The resources provide the stakeholders with a way

to interact with the system-of-systems. Additionally, the stakeholders have a direct effect on the resources, as their use of the resources can affect their availability. In the case of the complete navigation system-of-systems, there are two major resource families: navigation systems and navigational pathways. The navigation systems can be broken down into two major subclasses, the transmitters and the receivers. The transmitters are the devices which provide the navigation signal. These can range from global positioning system (GPS) satellites to wireless internet access points to cell phone towers. The receivers are the devices which receive the signal and provide the user with the navigation instructions.

The other major class of resources is the navigational pathways. The pathways are the various means of travel; roads, paths, sidewalks, building corridors, etc. that allow a person to travel from one point to another. Additionally, the mode of travel is relevant, i.e. car, bus, foot, bike, etc. There pathways are subdivided: free flowing pathways, such as roads, sidewalks, hallways, and anywhere a stakeholder can travel freely; and scheduled pathways, best represented by railroads and flights where at specific times, there is a capacity to travel from one point to another.

3. Drivers may include economic disruptions which reduce the need for commercial navigation, as well as social disruptors causing both spikes and lulls in the social demands for a navigation system. For example, during Super Bowl Sunday, there would be an increased demand for navigation before and after the Super Bowl as people head to the location where they watch the football game and return home. However, once the game started, there would be a lull in demand, as many people would be watching the game.

   There are also multiple disruptors to the resources. One of the major disruptors is the loss of navigation system signal, which can be caused by either the failure of the transmitter, like a power outage, or the inability of the receiver

to receive the signal, like a GPS unit indoors. Competition for resources can also cause disruptions to a network. The best example of this would be traffic on a highway. However, this also applies to other modes of transportation as well. Additionally, some navigation systems, such as cellphones and wireless internet have limited capacities. If there are too many people using the system or obstructions to navigational pathways such as blocked or closed roads and locked buildings, failure could occur.

4. The disruptors are anything that can cause interference with either the drivers or the resources. There are likely multiple disruptors in an SoS.

	Resources	**Operations**	**Policy**	**Economics**
α	Navigation Device (e.g. RFID, Wireless, GPS, embedded sensors, IMU, etc)	Operating a resource (e.g. GPS navigation system, RFID navigation system)	Policies relating to single device use (e.g. Wireless – CDMA, GPS – GNSS)	Economics involved with one navigation device
β	Groups of navigation device assigned to serve a user	Operating a navigation device group to serve a user in optimizing path (e.g. group A consisting: user, RFID, GPS, IMU, etc)	Policies relating to navigation device group use (e.g. switching between devices based on location/availability)	Economics of navigation group device use (setup, operated, provide service)
γ	Regional (city or county) navigation system integrating other users	Operations of device groups simultaneously with other users (e.g. groups A, B and C)	Policies relating to regional navigation system (device group communication between users – "Navigation Protocol"; managing data traffic between communicating groups)	Economics of business sector (service to multiple users)
δ	National Navigation System integrating other users	Operation of multiple, interwoven device major groups	Policies on National navigation system (e.g. National Traffic Management – tied into specific groups, e.g. National Air Transportation System)	Economics of national navigation systems
ε	Global Navigation System integrating other users	Global Operations in world	Policies on Global Navigation system	Global economics of global navigation systems

Figure 3.12.: This figure shows the lexicon applied in this environment. This figure is adapted from the work by DeLaurentis and Callaway [72]

Three areas of uncertainty to be addressed within the model are:

1. The first major area of uncertainty was the stochastic model of navigation system availability. A stochastic model was chosen because it was felt to best represent real world behavior of actual navigation systems.

2. Additionally, the position of the various agents and the accuracy of those positions cannot be determined with absolute certainty. The number of users and the method of transportation used was an unknown variable.

3. Finally, the traffic on the navigational pathways was unknown. While traffic was not included as a variable, the model is flexible to allow growth. Therefore, it was included as one of the areas of uncertainty.

3.3.1 Methodology

Agent-based modeling was implemented using MatLab as the simulation's development platform. The model's methodology is described in-depth in the following chapter.

3.3.2 Design

The majority of research in navigation and localization focuses on individual technologies or a subset of hybrid technologies. These are typically categorized into distinct areas (e.g. robotics, assistive devices, navigation, asset tracking, gaming, etc). It can be subdivided further into indoor versus outdoor environments or by selected mode of transportation (e.g. walking, driving, public transportation, etc).

Currently individual localization is available based on a user's location, network coverage or mode of transportation. This model was developed as an autonomous system to combine disparate localization technologies into a combined solution. The potential benefit of the system is multi-fold. It would allow directional acuity to a person without prior knowledge of the area and possibly open up new areas to those with a disability. There is potential for this research to be used to develop

an assistive device by providing location information and guidance to the visually impaired user. Additionally, this research could be combined with environmental knowledge such as traffic information or emergency situations that would allow alternate routing possibilities. The greatest benefit is the potential to create and define a navigational model that easily allows the addition of new technologies or disparate systems.

By identifying and analyzing parts of the SoS, the research moved to the formulation of the problem. A consideration of the research was to begin with simple and individual scenarios to better understand external variables and influences. For the initial model, a small map of the Purdue University campus in West Lafayette, Indiana was used. This provided a discrete and manageable area to analyze the model developed.

The uncertainty mentioned before, combined with the emergent behavior that was expected when the multiple navigation systems and multiple users begin interacting with each other, led to the use of agent-based modeling (ABM) as the modeling method of choice. This is a result of ABM being able to handle large amounts of uncertainty and the appearance of emergent behaviors. It also allows for appropriate modeling of human and dynamic behavior.

3.3.3 Implementation

ABM was selected for the modeling phase of the research based on its capacity to handle, with relative ease, large amounts of uncertainty and the appearance of emergent behaviors. The latter was expected when the multiple navigation systems and multiple users interact. ABM allowed for appropriate modeling of human and dynamic behavior.

The initial computational model, M_I, was created using a basic path map representation, and implementation of simple scenarios to check for expected functionality. As a path-planning algorithm requires a path, a method was needed to translate

the simulation space into a usable space within MatLab. Initially, an extremely basic model was used with P showing the path space, 7 representing an intersection in the path where a decision is made regarding the route, and I represents ∞ which indicates no path available. This is shown in Table: 3.2.

Table 3.2: *Basic Model with Limited Data P = path, 7 = intersection, ∞ = inaccessible*

∞	P	P	P	P	P	∞
P	∞	∞	∞	∞	∞	P
7	P	P	P	P	P	7
P	∞	∞	∞	∞	∞	P
∞	P	P	P	P	P	∞

After verifying the simple path representation in Table: 3.2 in MatLab, a more complex representation was created by generating a Google Map of the Purdue Campus in West Lafayette, Indiana (Fig. 3.13). This map was used as a frame of reference for the path map created as a numbered grid, with minimal paths.

MatLab read the path map and collected coordinates at each cell along the grid. The pathway was split at the intersections and path points sorted by coordinate. This allowed the agents to correctly traverse each path in the proper path point sequence. Verification was performed to ensure the correct separation of pathways and path point coordinate sequencing.

The first computational model implemented only trivial scenarios to verify correct functionality. The model's assumptions are shown in Table 3.3.

The navigation system used GPS exclusively and assumed no signal loss. The model retrieved information from the path-planning agent and directed the agents from a fixed start point to a fixed destination using the shortest calculated path.

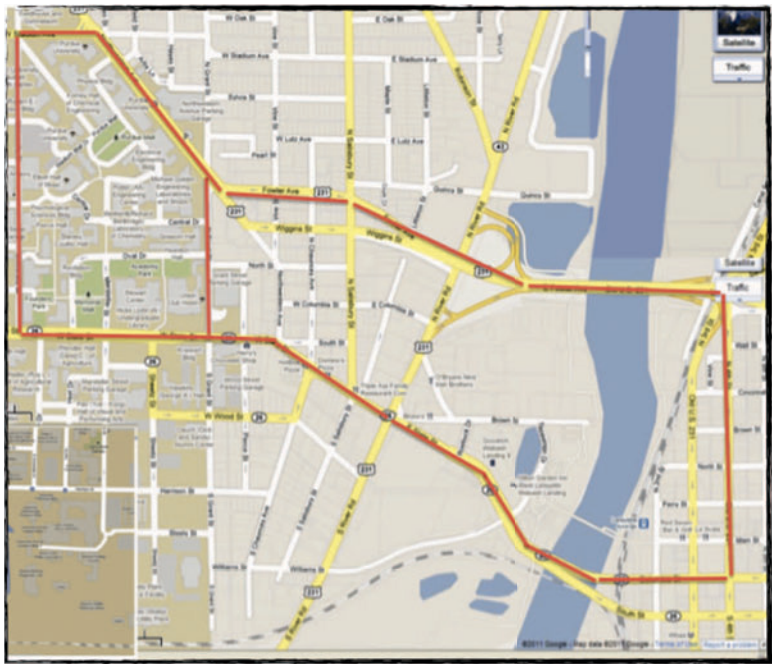


Figure 3.13.: Purdue Campus Map Red line indicates the MatLab path

Once this model was validated, the next version of the model was developed with additional specifications.

During development of M_{II}, the capability of the model was increased. This iteration included one or more users, a single travel mode, two navigation systems (GPS and WiFi), an increased number of paths, and stochastic variables. The assumptions are shown in Table: 3.4.

In this model, a cost map was created as a weighted adjacency matrix. Weights were assigned to each link (edge) in the path to assist in path-planning. The increase in complexity (with the inset legend Fig. 3.14) provided context to the information represented in the map. The model used the path-planning agents to calculate the cost of each pathway in order to provide the user agents the lowest cost route to the objective. To this end, the map provided cost information by type of pathway (e.g. highway versus side street) and mode of transportation (e.g. walking versus driving). This allowed the path-planning algorithm to attach meaningful significance when determining the lowest cost pathways for the user agent to travel.

Table 3.3: *Table of assumptions for M_I*

Single	user	agent
	objective	target
	speed	fixed
	sensor system	GPS
	mode of transporation	car
Fixed	start point	initial state
	end point	goal state
Objective	single agent	A[x,y] to B[x,y]
Constraints	agent stays on path	
	path is static	
	lowest cost path	weighted (0-10)
	signal strength	fixed and on(1)
	no sensor switching	GPS
	single transport mode	car
Variables	user location	[x,y]
	route cost	weighted

This model included GPS signal availability as a stochastic variable. Within range of Purdue campus buildings, the landscape was treated as an urban environment with the expectation that GPS signal loss could occur. This possibility of signal loss was calculated as a probability, within a radius of 7 cells, with a probability of 80%. The ability for the agent to utilize a Wifi network was included to buffer location information when GPS was unavailable. The model considered Wifi as having less complete overall information due to its lower effective range. This allowed the potential for the path to change depending on which signal was available to the user agent.

Table 3.4: *Table of assumptions for M_{II}*

Single	objective	goal
	mode of travel	car
Multiple	user	agent
	speed	sensor dependent (1-4)
	sensor system	GPS, Wifi
Fixed	start point	initial state
	end point	goal state
Objective	multiple agents	fixed
Constraints	agent stays on path	
	path may change	sensor dependent
	lowest cost path used	
	probabilistic sig strength	based on location (.80)
	sensor switching	GPS, Wifi
	single transportation	car
Variables	user location	[x,y]
	route cost	weighted
	probabilistic sig strength	based on location
	sensor system in use	GPS, Wifi

Again, once the experiment was validated, a more complex model was implemented, M_{III}. The experimental setup for M_{III} contained the greatest number of characteristics and variables. The goal was to approximate a full representation of a street map of the Purdue area (Fig. 3.15). The coordinates were converted into a discrete set of nodes with adjacency matrices representing pathways which the agents could travel (Fig: 3.16). The assumptions for this model are shown in Table: 3.5.

When GPS was available, user agents traveled at their fastest speed along their planned path, as they had full information. If GPS was lost but Wifi was still

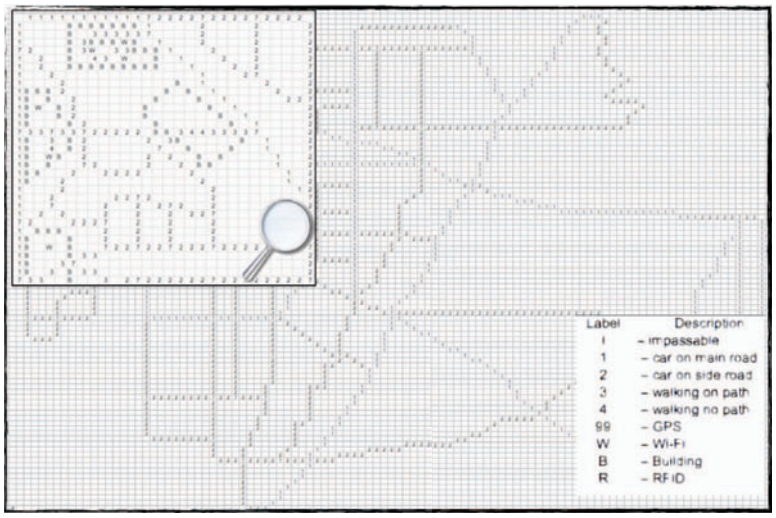


Figure 3.14.: Cost Map

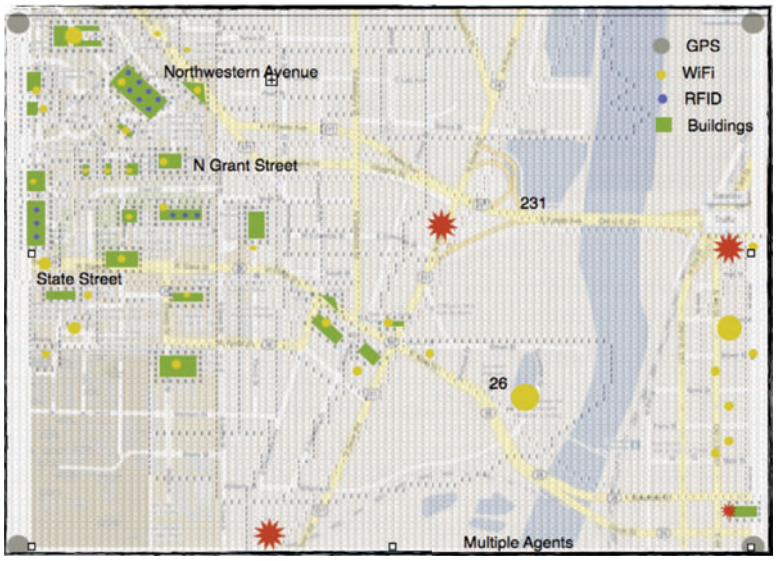


Figure 3.15.: Complex Cost Map

available, they traveled at a slower speed since Wifi does not provide position data as completely as GPS. In order to minimize route loss, the agents with only Wifi would follow a path which emphasized maintaining a Wifi connection until reaching the destination. If neither GPS or Wifi were available, the user agent would move at the slowest speed along its last known path until navigation signals could be reacquired.

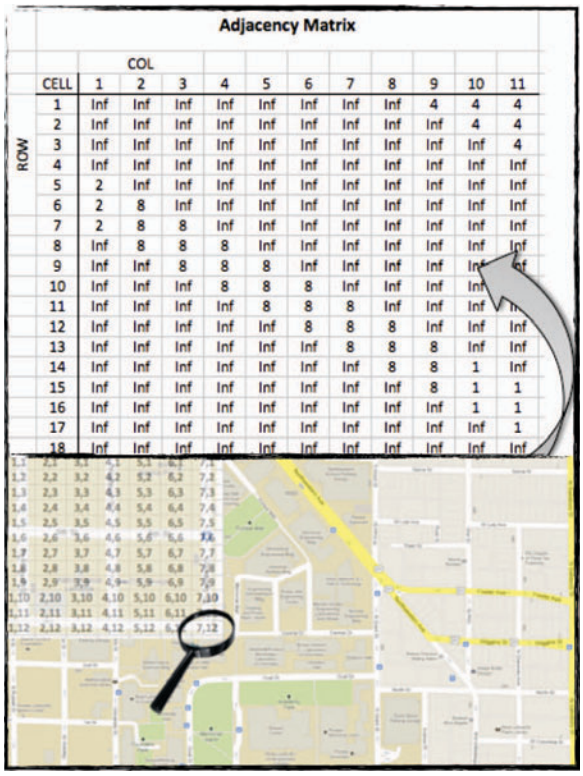


Figure 3.16.: Adjacency Matrix

GPS is assumed to be available across the entire map, however near buildings GPS was modeled with the possibility of signal loss or multi-path loss. This was modeled as a stochastic variable where an agent next to a building has an 80% chance of signal loss, but that probability decreased linearly to zero as the distance increased. With Wifi, the availability depended on the distribution of Wifi sources on the map. If an agent was within the communication range of the Wifi node it would have connection, however there was still a 30% chance of a dropped signal when within range.

The agents used were: user agents, target agents, and path-planning agents. The user and target agents were randomly selected from a discrete set of possibilities. The goal of the user agent was to follow the shortest possible path using Dijkstra's algorithm. The user agent would use the target agent's broadcasted position to intercept the target. Dijkstra's algorithm looks for the total minimum cost pathway;

Table 3.5: *Table of assumptions for M_{III}*

Multiple	user	agents
	objective	target
	mode of travel	drive, walk
	speed	sensor dependent
	sensor system	GPS, Wifi, RFID
Discrete Set	start point	quasi-random
	end point	goal state
Objective	multiple agents	multiple A to B
Constraints	path may change	sensor dependent
	lowest cost path used	weighted
	probabilistic sig strength	based on location
	sensor switching	GPS, Wifi, RFID
	multiple transport modes	car, walk
Variables	user location	[x,y]
	route cost	weighted
	probabilistic sig strength	
	sensor system in use	GPS, Wifi, RFID

the cost is the sum of the values assigned to the nodes in the Cost Map for all nodes in the path. The path was stored in the user agents' cache and was only recalculated if their navigation system status changed (i.e. GPS, Wifi or RFID signal was lost or gained). The primary goal was to travel to (or intercept if moving) the target unless or until GPS signal was lost. If GPS was lost, and Wifi was available, the end goal remained the target, however a higher priority interim goal was to stay in Wifi range until GPS is regained. If both systems were unavailable, the agent would check for RFID and use that system to correct any localization discrepancies. The agent

remained on its last known course, at a slower signal, until any further signals are regained.

Targets were chosen randomly from a discrete set of predetermined locations. The target agents were designed to move to a new destination restricted to the academic campus every so often, and would update their current positions via broadcast message. These were available to the user agents, who would change their route accordingly.

3.3.4 Results

During each run, data as a simulation statistic based on x and y position, GPS, Wifi and RFID signal availability, and speed, were recorded by time-step (Fig. 3.17). A more detailed look at the same data separated by individual run is shown in Fig. 3.18.

X	Y	GPS	Wifi	Speed
33	96	1	0	0
37	93	1	0	4
41	88	1	0	4
43	83	1	0	4
46	78	1	0	4
48	73	1	1	4
50	68	0	1	4
51	63	1	1	2
51	59	0	1	4
48	57	1	1	2
47	52	1	0	4
47	47	1	1	4
47	42	1	1	4
47	37	1	1	4
47	32	1	1	4
42	31	1	1	4
37	31	1	1	4
32	31	0	0	4
29	26	1	1	1
25	21	0	1	4
23	16	0	1	2
18	11	0	1	2
14	6	0	1	2
11	5	0	1	2
9	5	0	1	2

Figure 3.17.: Example of Simulation by Time-Step

[h!]

	RUN 1					
X	33	33	33	33	33	33
Y	96	96	96	96	96	96
GPS	1	1	1	1	1	1
WIFI	0	0	0	0	0	0
SPEED	0	4	4	4	4	4
	RUN 2					
X	33	33	33	33	33	33
Y	96	96	96	96	96	96
GPS	1	1	1	1	1	1
WIFI	0	0	0	0	0	0
SPEED	0	4	4	4	4	4
	RUN 3					
X	100	100	100	100	100	100
Y	86	86	86	86	86	86
GPS	1	0	0	1	0	0
WIFI	0	0	0	0	0	0
SPEED	0	4	1	1	4	1
	RUN 4					
X	101	101	101	101	101	101
Y	41	41	41	41	41	41
GPS	1	1	1	1	1	1
WIFI	1	1	1	1	1	1
SPEED	0	4	4	4	4	4

Figure 3.18.: Sample of Results Recorded by Individual Run

Several components varied in each iteration of the model. Pathways between deliverer and target agents were increasingly complex. Changes in signal strength and signal loss along the route were incorporated. By gradually increasing the representation of the map, buildings and varied types of movement over the environment were included. These were shown by different speeds based on mode of transportation and size or type of pathway.

Additionally, the distribution and density of Wifi nodes throughout the map was varied. The model used Wifi as a backup to GPS, with GPS having the probability of signal loss only in proximity to buildings. The test was to determine if the addition of Wifi nodes to the map allowed for faster and more reliable travel for the agents.

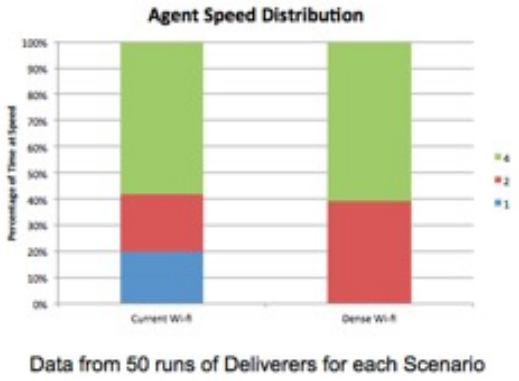


Figure 3.19.: Agent Speed Distribution

In running the model over time it was found that the addition of only the Wifi system did not generate a large performance gain in the agents' behavior. Often GPS alone was adequate, by just continuing along the known route until GPS was available. Adjusting the probabilities that the GPS and Wifi signals would be lost, but additional Wifi nodes were found, made an improvement of just under 20% change in speed (Fig. 3.19). The GPS probability of loss was 80% next to buildings, 0% when 7 units from buildings, and Wifi experienced no signal loss when within range.

Adding a stochastic element to the Wifi or adjusting GPS multi-path loss created limited improvements for the agents.

These results were recorded with no assumptions on performance of the devices used for such a system (i.e. Wifi triangulation is much less battery and computationally intensive than GPS). The path-planning and agent movement was only shown based on sensor availability. The results to date have shown that in the most GPS-restrictive environments, such as very dense urban environments with large buildings or natural restrictions like canyons, the navigational improvement due to addition of available Wifi is minimal. This lines up with the goal of future work which includes adding new individual systems and sensors to the simulation for improvement, as well as identifying reliability measurements that can be compared between the systems. Additionally, incorporating movement between indoor and outdoor environments will allow for better results.

3.4 Conclusion

Table 6.1 presents the prototypes and the results from the prototypes, as follows:

Table 3.6: *Comparison of Prototypes*

Prototype	P_I	P_{II}	P_{III}
Platform	Samsung Nexus S	Apple iPhone	Matlab Simulation
Domain	Path-plan, nav	Path-plan	Path-plan, nav, localize
Method	Experiment	Experiment, MAS	MaS predicated on DEVS
Results	Integration of GPS, Wifi and RFID	Successful ABM, Embedded sensors too noisy	Successful MaSoS

CHAPTER 4. METHODOLOGY

This research melds existing sensor technologies, with hand-held hardware and software to enhance personal navigation through a densely trafficked urban environment. The research methodology of the multi-agent system-of-systems (MaSoS) and the quantitative research design guided the development of experiments as detailed in this chapter. Agent-based modeling was examined in the context of a MaSoS and pointed to the research landmarks which were available to shape the experimental effort as it moved forward.

Discrete event-simulation (DEVS) is introduced and formally defined. The methodology of DEVS (and specific permutations such as Z-DEVS) is used to convert the formal specifications into logical (i.e. code supporting) specifications [49]. Path-planning (through Dijkstra's algorithm and the Nearest Neighbor algorithm) is introduced; these serve the needs of the model created on which the simulations were run.

This research is novel as it fuses existing sensor technologies and a framework to easily allow the addition of new or expanded technologies. Simulations are implemented as agent-based models, while the prototypes described are developed as practical applications of the research. The methodological approaches presented replicate the research goal to achieve a previously-unexploited deployment of sensor technologies within a methodological framework. In the future, this framework will more readily enable testing of new or additional technologies. The refinement of this methodology is itself an advance in the technique of agent-based modeling within a system-of-systems environment.

The ultimate goal of the research design is to determine and quantify the relationships between the variables which were tested in the simulations. The steps used

to develop the methodology were broken into three categories: concept, description and refinement.

During the *concept phase*:

1. Data was collected about the specific physical environment (e.g. Purdue University campus);
2. Selection of the prototype hardware, the Samsung Nexus S phone, running the Android operating system;
3. Development of the prototype Java-based navigation software, to run on the Android platform;
4. Preliminary understanding of constraints of this code-writing effort; and
5. Identify possible issues in the hardware and software considered in isolation from the physical environment selected.

In the *description phase* and during the formal specification, assumptions, constraints, redundancies and omissions were identified and corrected. Consideration was then given to an iterative process to repeat the last steps as necessary.

The *refinement phase* included:

1. Identify optimal, efficient and low cost results at the end of each experiment.
2. Assess results from experiment with prototype in the selected physical environment.
3. Refine concepts in Java software to enable developments in the MatLab-based simulations.

The five objectives in developing the methodology were:

1. Conceptualize (through research in the literature, Chapter 2) the patterns identified which would yield optimal results. It is important to note that this does not mean 'success on the first try', but rather an advance which yields an alternative route for further experimentation.

2. Extract operational definitions from the literature for each key component in the hardware and software.

3. Utilize established protocols for writing formal specifications of code, identifying logical opportunities, difficulties and assumptions in the code writing effort.

4. Construct test scenarios to validate each experiment with the chosen prototype hardware and software.

5. Identify the modeling prototype that optimizes performance given the problems and opportunities presented with hardware and software similar to the field prototypes.

A critical and quantitative examination of the appropriateness of an agent-based model for a navigation application in a complex campus environment is explored. A formalized description of the system and the model are determined. An agent-based model is created within an SoS to allow for the addition of new technologies. This research quantifies the experiments by measuring:

1. which system is active and providing sensor information;

2. the speed allowed by that system;

3. the inclusiveness of each sensor system (e.g. indoor and/or outdoor).

Bonabeau [113] identifies several situations in which agent-based modeling is useful:

- interactions between the agents are complex, nonlinear, discontinuous, or discrete; this research has discrete interactions and is run by time-step;

- space is crucial and the agents' positions are not fixed;

- the population is heterogeneous, (i.e. each individual is potentially different);

- the topology of the interactions is heterogeneous and complex; the topology in the simulation is designed to emulate a complex urban environment;

- the agents exhibit complex behavior, including learning and adaptation.

Agent-based modeling is therefore selected, as the defined problem involves decentralized autonomous nodes (users), self-organized links (navigational pathways), and human behavior.

4.1 Model

Agent-based modeling is a form of computational modeling. The goal of ABM is to describe the evolution of a dynamic system and simulate the behavior of its agents. To this end, Schank believes that agents, their environment and rules for their behavior are represented in a computer program [116]. Simulation applications are typically written in an object-oriented language (such as Java) as these types of languages have many of the required concepts (e.g. inheritance). The simulation is able to observe the evolution of agent and agent interactions over time with a minimum of initial assumptions.

Due to the ability to change input elements, as well as rules, location and time variables, the simulation will necessarily be run multiple times. Sequential runs allow for the calculation of both mean and variance, and the possibility of the data converging on expected values. This research has been implemented and prototyped over a variety of platforms: Objective-C, Java and MatLab.

As initially organized the agent-based model defines the individual agents and their internal structure; and identifies rules for acting and interacting between agents. To define the individual agents programmatically, there is a class of agents which includes an appropriate data-structure, methods to manipulate information and define interactions, and communication between agents and the environment.

ABM offers a computational toolkit for developing precise and specific models of how individuals interact, and for discovering patterns of behavior and organization that emerge from these interactions. ABM forces the user to be explicit and specific about the assumptions made. The art of ABM is finding assumptions that introduce

relatively little bias and error. This activity is not predicated on specific rules, and general experience can be helpful.

There are a variety of known limitations or weaknesses associated with agent-based modeling. According to Bonabeau et al., "a model has to serve a purpose; a general-purpose model cannot work. The model has to be built at the right level of description, with just the right amount of detail to serve its purpose [113]. To achieve this properly or efficiently can be difficult. It is also limited by the completeness of knowledge of the modeler and information about the event being modeled. Additionally, the output of the model must be analyzed. Inexperience and incomplete inputs can lead to incorrect interpretations of the output.

Limitations in using ABM occur when modeling human agents and their behavior. Human agents are complex, with the potential to be irrational and subjective in their decisions. It can be impossible to predict irrational behavior or decisions. Fortunately, by running a simulation multiple times, it is possible to see emergent outcomes based on many different interactions. Another issue with ABM: as the size and complexity of the model and number of agents grows, the modeling can become computationally- and time-intensive.

ABM is inherently suited to complex systems or system-of-system issues as it is a "flexible tool for analysis of large-scale interacting systems [14]. According to Zhao and DeLaurentis, ABM 'allows for the most natural descriptions of interactions among multiple, independent and heterogeneous entities [14]. This is especially critical as the observable output of emergent behavior is often unexpected and counterintuitive.

ABM utilizes simple rules to control agent behavior. Agents can have many properties: asynchronous; dynamic; independent; active; flexible; self-contained or adaptive. One of the popular features of an ABM is its ability to use simple rules to generate complex and emergent behavior. According to Bauer et al., the computational structure is parallel and is able to be implemented efficiently on parallel computers [43]. Some of the observable behaviors are periods and patterns (spatial and temporal).

According to Macal and North [42] the popularity of agent-based modeling can be attributed to 3 reasons:

1. the increasing need to model complex things and their interdependencies;
2. greater levels of detail are available; and
3. computational power is advancing rapidly enough to support more complex simulations.

Although the *rules* to control agent behavior can be simple, ABM is capable of handling agent interactions that are complex, nonlinear, discontinuous or discrete. It is also useful when space is limited or agent position is not fixed. If each agent is individualized, or the agent behavior is adaptable through changes, memory or learning, then agent-based modeling can be appropriate.

4.1.1 Principles

The steps to build an ABM include:

- identify the agents;
- define the rules governing interactions; and
- determine the appropriate time and space.

When identifying the agents, they should have the following properties: internal states; ability to interact with the environment; access to shared knowledge; dynamic; and specific levels of interaction (i.e. global or local).

The most useful principle of ABM is the use of simple rules (as opposed to mathematical equation modeling) that can result in complex and emergent behavior during the simulation. Examples of simple rules used in this research are:

- if the agent is not located in the same building as the goal point, the nearest exit is found;

- if a grid cell contains an obstacle (e.g. a building), the agent must plan a route around it; or

- if an agent is within a specified range of a building, then with a certain probability P_x, GPS becomes unavailable at that location.

The agents modeled are dynamic and able to create real world complexity, which is difficult to do using only mathematical equations. The agents are also discrete time events, situated in a specific time and space. Being observed at a discrete time slice allows the same agent to exhibit different or altered behavior simply by observing it at another time interval or location. ABM is able to provide an indication of the system's robustness as it observes agents adapting or changing to handle internal or external changes. Characteristics of the analytical agent-based model are shown in Table: 4.1. This table identifies the Objects, Variables, States and Dynamics of the agents and details valid ranges of values for each. A cross-referenced notation of the associated DEVS formalism is also noted. DEVS is formalized in Section: 4.2.

The scope of the characteristics are also included in the table. The variables are position, signal, path and cost. The position of the agent is given by the Cartesian coordinate [x,y]. The signal is a binary value and is 0 for off, or 1 for on. The predicated path is a Cartesian coordinate [x,y] of the next step in the path. The cost function is a minimalization function between 1 and 10, with 1 indicating the lowest cost and 10 the highest.

The states are determined by time-step, transportation method, active sensor system and speed. These are also defined with a range of values. The time-step is given as a count of the time of each state. The transportation method is a binary value, 0 for car, 1 for walk. The active sensor is a numeric value; 0 for GPS, 1 for Wifi, or 2 for RFID.

Finally, the dynamics of the system are defined. The available dynamics are a change of location, travel speed, travel mode, sensor system in use or objective. For a change of location, the value is expressed as a Cartesian coordinate; [x,y]. A

Table 4.1: *Table of Characteristics*

ABM	DEVS	Scope
Objects	*Model*	*Value*
. agents	M	
Variables	*Model*	*Value*
. position of the agent	X, Y	node[x,y]
. signal strength	S	[0,1]
. predicted path		node[x,y]
. cost function		[1:3=car, 4:6=walk on path 8:10=walk off path]
. GPS probabilistic availability		[60%, 80%]
States	*Model*	*Value*
. time-step	$\tau\psi$	single step = 1
. transportation method		[0=car,1=walk]
. active sensor system	δ_{ext}	[0=GPS,1=Wifi,2=RFID]
. speed	δ_{ext}	[1:3=car, 4:5=bike, 6=run 7:10 = walk]
Dynamics	*Model*	*Value*
. change of location		node[x,y]
. change of travel speed	δ_{ext}	[1:10]
. change of travel mode		[0=car,1=walk]
. sensor system in use	S	0=GPS,1=Wifi, 2=RFID, 3=na
. objective	Y	Goal[x,y] and 0 = not there, 1 = goal

change in travel speed is a range given 1 through 10. This is also a minimalization function with the highest speed being the lowest value. According to a study on the average walking speed [130], humans walk approximately 3.5 miles per hour. The posted speed limit throughout the majority of the Purdue Campus is approximately 25 miles per hour. Converting this to a 1-10 scale with faster speeds being shown as smaller numbers, the scheme is: 1 is driving, 5 is running or biking, 4 is walking outside, 3 and 2 are walking inside, and 1 is slowly moving. The change in travel mode is a binary value with 0 for car, 1 for walking. The sensor system deployed

relies on combinations of available GPS, Wifi or RFID. The final dynamic is the goal. This is a Cartesian coordinate [x,y] and the indication that an agent has reached it is given in binary 0 for not complete and 1 for complete.

The agent behavior is modifiable by defining the rules of interaction. As explained below, it is expedient to limit potential agent interactions, by defining these interactions to a states with discrete events providing transitions to new states. This approach allows many types of interaction to be observed; emergent behavior in particular can be recognized as different values are assigned to the relevant variables.

Bonabeau et al. [113] has identified three benefits of agent-based modeling:

- *Emergent Systems*: Agent-based modeling is able to capture emergent phenomena due to the interactions between individual agents. The result is often a complex property that is difficult to predict or even understand. An example provided by Bonabeau et al., is a traffic jam that results from interactions between drivers, but can cause a reaction in the traffic moving in the opposite direction.

- *Natural System Description*: Providing an easier and more complete means to describe agents is an additional benefit. For novice or inexperienced modelers, describing the movement of an agent can be more natural and understandable than detailing specific equations for dynamic movement.

- *Flexibility*: The benefit of flexibility is multi-pronged. It is easy to add or remove agents from an agent-based model, as each agent is individual and autonomous. Additionally, by using independent agents, it is easy to change the behavior, description, grouping and aggregation to see what the result would be, especially advantageous if the best configuration is not known.

4.2 Implementation

To begin, a simple scenario was modeled to better understand external variables and influences. For the initial model, a small map of the Purdue University campus in West Lafayette, Indiana was used, which provided a discrete and manageable area to analyze.

Agent-based modeling was implemented using MatLab. In the work by Onggo [50], a formalized ABM is defined as:

Definition 4.2.1 *A tuple* $< A, E >$*, where* A *is a set of agents* ($A = \cup a^k$ *and* $1 \leq k \leq N_{agents}$,*) and* E *is the environment.*

The simulation was broken into four iterative computational models for recurring verification of functionality:

- M_I a simple computational model with limited variability used to initially verify the simulation;
- M_{II} an intermediate computational model developed with increased complexity and the addition of more paths and single variables;
- M_{III} a more complex computational model with the greatest number of agents, variables and paths, representing a full street map of the Purdue Campus;
- M_{IV} a single agent simulation moving from the second floor of one building to the first floor, then outside and across campus and to another building before moving up to the second floor. The characteristics of the system model are that it is a discrete event simulation. It has stochastic parameters and runs in a dynamic environment. Initially, the environment is partially-observable and is contained within a finite state space.

The agent-model is constructed as a discrete event simulation (DEVS) using the formal model introduced by Bernard Zeigler [48]. The purpose of the discrete-event simulation is that the model executes on events instead of simply on time. This

allows the model to update only when a system state change has occurred, which saves computation time. The model is created as an atomic two-dimensional grid of rectangular cells. It is formalized by Zeigler as:

Definition 4.2.2

$$M = (X, Y, S, \tau, \delta_{int}, \delta_{ext}, \lambda), \tag{4.1}$$

where M is the model defined by,

- *X is the set of input events;*
- *Y is the set of output events;*
- *S is the set of sequential state values;*
- *$\tau\psi$is the time advance function used to determine the time of a state (e.g. $S \rightarrow \mathbb{T}^{\infty}$);*
- *δ_{int} is the internal transition function of the system state (e.g. $S \rightarrow S$);*
- *δ_{ext} is the external transition function from the input event (e.g. $QxX \rightarrow S$);*
- *λ is the output function based on the system state (e.g. $S \rightarrow Y^{\phi}$);*

The model can be described as having an input (X) and output (Y), in this research this is represented by the Cartesian coordinates of the start and end points (CC_{start}, CC_{end}). These are the means of communication with other models. Every state (S) is associated with a time advance function (τ), which determines the length of the state time. This model has a number of states (e.g. $GPS_{on}, GPS_{off}, Wifi_{on}, Wifi_{off}, RFID_{on}, RFID_{off}, Signal, NoSignal, CostPenalty$, $Speed_{change}$). When the transition time ends, the execution results are sent using the output function (λ). The call to the internal transition function (δ_{int}) is made and a state change occurs. Input events are collected and the external transition function (δ_{ext}) indicates the appropriate response.

The model for the research can be represented with this formalism as:

- X =$\{\mathrm{CC}_{start}\}$;
- Y =$\{\mathrm{CC}_{end}\}$;
- S = $\{\mathrm{GPS}_{on}, GPS_{off}, Wifi_{on}, Wifi_{off}, RFID_{on}, RFID_{off}, Signal, NoSignal, CostPenalty, Speed_{change}\}$;
- $\tau(S) = 2s, \sum \tau(S)_{since} \Delta S$;
- $\delta_{ext}(GPS_{on}) = GPS_{off}$;
- $\delta_{ext}(GPS_{off}) = GPS_{on}$;
- $\delta_{ext}(Wifi_{on}) = Wifi_{off}$;
- $\delta_{ext}(Wifi_{off}) = Wifi_{on}$;
- $\delta_{ext}(RFID_{on}) = RFID_{off}$;
- $\delta_{ext}(RFID_{off}) = RFID_{on}$;
- $\delta_{ext}(Speed_{x_1}) = Speed_{x_2}$;
- $\delta_{ext}(Speed_{x_2}) = Speed_{x_1}$;
- $\delta_{ext}(Signal) = NoSignal$;
- $\delta_{ext}(NoSignal) = CostPenalty$;
- $\delta_{ext}(CostPenalty) = Signal$;
- $(GPS_{on}), (GPS_{off}), (Wifi_{on}), (Wifi_{off}), (RFID_{on}), (RFID_{off}), (Speed_{x_1}), (Speed_{x_2}), (Signal), (NoSignal), (CostPenalty) = CC_{end}$;

Additionally, path-planning is done using Dijkstra's algorithm for finding the single-source shortest path [129]. This is formalized by Bang-Jensen and Gutin [131] as shown :

Definition 4.2.3 *Input: A weighted digraph $D = (V, A, c)$, such that $c(a) \geq 0$ for every $a \in A$, and a vertex $s \in V$.*
Output: The parameter δ_v for every $v \in V$ such that $\delta_v = dist(s, v)$.

- *Set $P := \theta$, $Q := V$, $\delta_s := 0$ and $\delta_v := \infty$ for every $v \in V \quad s$.*
- *While Q is not empty do the following:*
  - *Find a vertex $v \in Q$ such that $\delta_v = min\{\delta_u : u \in Q\}$.*
  - *Set $Q := Q \quad v, P := P \cup v$.*
  - *$\delta_u := min\{\delta_u, \delta_v + c(v, u)\}$ for every $u \in Q \cap N + (v)$. ([129]).*

To prove the correctness of Dijkstra's algorithm, it suffices to show that the following proposition holds.
Proposition:
At any time during the execution of the algorithm, we have that:

- (a) For every $v \in P$,$\delta_v = dist(s, v)$.
- (b) For every $u \in Q$,δ_u is the distance from s to u in the subdigraph of D induced by $P \cup u$.

Proof *When $P = \theta_s = dist(s, s) = 0$ and the estimates $\delta_u = \infty, u \in V \quad s$, are also correct.*

Assume that $P = P_0$ and $Q = Q_0$ are such that the statement of this proposition holds. If $Q_0 = \theta$, we are done. Otherwise, let v be the next vertex chosen by the algorithm. Since Part (b) follows from Part (a) and the way in which we update δ_u in the algorithm, it suffices to prove Part(a) only. Suppose that (a) does not hold for $P = P_0 \cup v$. This means that $\delta_v > dist(s, v)$. Let W be a shortest (s, v) - path in D. Since $\delta_v > dist(s, v), W$ must contain at least one vertex from $Q = Q_0 \quad v$. Let u be the first vertex on W which is not in P_0. Clearly, $u \neq v$. By the Proposition and the fact that $u \in W$, we have $dist(s, u) \leq dist(s, v)$. Furthermore,

since the statement of this proposition holds for P_0 and Q_0, we obtain that $dist(s,u) = \delta_u$. This implies that $\delta_u = dist(s,u) \leq dist(s,v) < \delta_v$. In particular, $\delta_u < \delta_v$, which contradicts the choice of v by the algorithm.

4.2.1 Conceptual Description

Research in navigation and localization often focuses on individual technologies or a subset of hybrid technologies. These are typically categorized into distinct usage areas (e.g. robotics, assistive devices, navigation, asset tracking, gaming, etc). They can be subdivided further into indoor versus outdoor environments or by selected mode of transportation (e.g. walking, driving, public transportation, etc). Individual localization is available based on a user's location, network coverage or mode of transportation.

The goal of the model is to combine disparate localization technologies into a uniform solution. There are many potential benefits: allowing directional acuity to a person without prior knowledge of the area; opening unknown areas to those with a disability; developing an assistive device by providing location information and guidance to the visually impaired user; combining the research with environmental knowledge such as traffic information or emergency situations that would allow alternate routing possibilities; and the possibility of creating and defining a navigational model that easily allows the addition of new technologies or disparate systems.

The first major area of uncertainty is the stochastic model of navigation system availability. A stochastic model was chosen because it was felt to best represent real world behavior of actual navigation systems. Additionally, the position of the various agents and the accuracy of those positions cannot be determined with absolute certainty. The number of users and the method of transportation of these users is an unknown variable. Finally, the traffic on the navigational pathways is unknown. While traffic was not included in the models developed for this research, the models

were intended to be flexible and to allow growth. Therefore, it is included as one of the areas of uncertainty.

The uncertainty mentioned, combined with the emergent behavior that is expected when the multiple navigation systems and multiple users begin interacting with each other, led to using agent-based modeling (ABM) as the modeling method of choice. This is a result of ABM being able to handle large amounts of uncertainty and the appearance of emergent behaviors. It also allows for appropriate modeling of human and dynamic behavior.

4.3 Data Collection

Data collection occurs during the modeling and simulations. The data is discrete and will be measured over time intervals from path start to finish. The data collection will be done by logging the state of the agents at each time interval. Other simulation details will also be logged (number of Wifi nodes and the range, number of RFID nodes and the range, stochastic value of GPS and the total cost of the path). A sample of a single agent is shown in Table: 5.4. In this table a small 6 step time sequence is shown. Each t_x is equal to one tic. The first quintet identifies the starting position of the agent. It shows if the agent is inside or outside, the Cartesian coordinates x and y, the floor of the building (if the agent is inside) and the building of origin (Knoy, Lawson or outside). The second quintet shows the same information, but indicates the current position for each time-step. The third quintet shows the information for an intermediate destination. If the goal is different from the start position (e.g. start is inside, goal is outside; or start building is Knoy, goal is Lawson). The goal quintet is for the final destination information. State 21 shows the cost of the path step, which is read in from the two-dimensional grid. The goal state (step 22) is a binary value indicating if the goal has been reached. Step 23, 24 and 25 show the signal for each of the sensors (0 for no signal, 1 for signal). Step 26 is the speed which is dependent on the location and sensor system, while step 28 collects penalties

incurred if there is no signal. Finally, step 29 totals the cost of each time-step by adding step 21, 26 and 28.

Table 4.2: *A sampling of the state of an agent over a time-step*

	Single Agent Object	t_1	t_2	t_3	t_4	t_5	t_6
1	Start In/Out	1	1	1	1	1	1
2	Start X	42	42	42	42	42	42
3	Start Y	107	107	107	107	107	107
4	Start Floor	2	2	2	2	2	2
5	Start Building	1	1	1	1	1	1
6	Current In/Out	1	1	1	1	1	1
7	Current X	42	41	40	39	38	37
8	Current Y	107	107	107	108	109	108
9	Current Floor	2	2	2	2	2	2
10	Current Building	1	1	1	1	1	1
11	Intermediate In/Out	1	1	1	1	1	1
12	Intermediate X	11	11	11	11	11	11
13	Intermediate Y	75	75	75	75	75	75
14	Intermediate Floor	2	2	2	2	2	2
15	Intermediate Building	1	1	1	1	1	1
16	Goal In/Out	1	1	1	1	1	1
17	Goal X	41	41	41	41	41	41
18	Goal Y	61	61	61	61	61	61
19	Goal Floor	2	2	2	2	2	2
20	Goal Building	2	2	2	2	2	2
21	Path Cost	2	2	2	2	2	2
22	Goal State	0	0	0	0	0	0
23	GPS	0	0	0	0	0	0
24	Wifi	1	1	1	0	0	0
25	RFID	0	0	0	0	0	0
26	Speed	8	8	8	10	10	10
27	Index	0	0	0	0	0	0

Continued on next page

Table 4.2: *(Continued) A sampling of the state of an agent over a time-step*

	Object	t_1	t_2	t_3	t_4	t_5	t_6
28	Penalty	0	0	0	2	4	6
29	Total Step Cost	0	14	14	18	18	20

4.4 Data Analysis

The data collected in the experiment is quantitative. This makes the data suitable for analysis using a variety of statistical methods discussed below. The data will be organized and grouped to make it more easily understandable.

4.4.1 Statistical Methods

Empirical research is a means of finding knowledge through direct observation. The evidence or data gained by recording these experiences can then be analyzed. An accurate analysis of the data is required to determine the probable validity of the research and acceptance of the initial research hypothesis.

Descriptive statistics are used to observe the patterns in the data. Histograms will be observed to visualize the data. Pearson's Chi-Square Test of Independence will be used to test the null hypothesis with the more rigorous of the statistical values 0.01%. Additionally, the mean, standard deviations and percentage of availability is studied.

4.4.2 Reconciliation

Following and extending the measures in Kim et al. [94], the metrics used to evaluate the model are divided into three categories: signal detection and coverage; independent network reliability; and overall navigation time.

The proposed system relies on the availability of individual navigation systems and the ability to move among them as appropriate, independent of user action. To use these systems in a beneficial way, data must be collected for several of their properties. Availability of the systems must be investigated; this includes geographical availability, user equipment necessary to take advantage of the system, and frequency of updates. The accuracy of these systems must also be known in order for prioritization, merging, and error correction of position information obtained by users. Level of accuracy will be explored for geographical location and/or distance from navigation system as well as expected error in sensor measurements (including drift in some systems).

4.5 Summary

This chapter presented the proposed methodology, model and description for this research. It offers both theoretical and practical discussion on the model and suggests to reconcile the difference during data collection. The methodology will include agent-based modeling and simulation, as described both formally and semantically. The next chapter focuses on the practical application of various systems to focus and guide the research.

This research is novel as it looks at the fusion of existing sensor technologies and creating a framework to allow the testing of new or additional technologies. The development of simulations and prototypes assisted in measuring the validity of the study. Simulations were then implemented as agent-based models, while two prototypes are developed as practical applications of this research.

The research expanded and refined the methodology as the prototypes were developed and tested. The methodological approaches detailed in this chapter seek to achieve a previously unexploited deployment of existing sensor technologies within a methodological framework that will, in the future, more readily enable testing of new

or additional technologies. Human localization and navigation offer rich opportunities for research, as a framework for further progress in this filed has not been established.

CHAPTER 5. AGENT-BASED SOS RESULTS

The results of the most recent simulations, run on M_{IV}, a revised model, are presented in this chapter. As previously explained, to isolate the critical variables , selected patterns of data configurations were employed. An extraction of the configurations are shown in Table: 5.1, with the majority omitted for brevity. The full table can be found in Appendix D, with the number of total simulations runs at over 14,000. The patterns derived from:

1. the probability of GPS loss (within a fixed distance of a building) at 60% and 80%;

2. a range of Wifi sensor which varied from 5, 10 or 15 cells from the node's location. The side of each cell or 'square in the virtual grid for indoor is approximately 2 meters, while the outdoor grid has cells with each side approximately 8 meters in length. (As used in this chapter, a cell is also a measure of signal strength: a signal of 5 cells is available to any reading device within a circle the radius of which, indoors, is approximately 5 virtual cell widths or 10 meters.)

3. the number of sensors distributed randomly on the virtual grid was set at 0, 10, 20 or 40 for both Wifi and RFID.

Each simulation pattern was run at least 100 times to allow the data to normalize. The complexity and level of uncertainty in the previous model (M_{III}) required simplification; this was designed into the latest model to minimize the noise, with the added benefit of providing sharper contrasts to choices between physical placement of nodes (location and number), and range of Wifi and RFID signal picked up.

In this experiment (M_{IV}), an agent was observed moving from a fixed start position to a fixed end position. The path was calculated for each portion of the

Table 5.1: *Extraction of Patterns of Variables*

Variable Node Count	GPS	Wifi Range	RFID Range	No Runs
Wifi-RFID				
00-00	60%	5	5	200
00-00	60%	10	5	200
00-00	60%	15	5	200
00-00	80%	5	5	200
00-00	80%	10	5	200
00-00	80%	15	5	200
00-10	60%	5	5	100
00-10	60%	10	5	100
00-10	60%	15	5	100
00-10	80%	5	5	100
00-10	80%	10	5	100
00-10	80%	15	5	100

simulation, but remained unaffected by signal availability. If the signal was lost and no other sensor system was available a cost penalty was incurred, but the agent continued along the predetermined path. The simulation data is recorded for each run and agent state, and then analyzed and presented later in this chapter.

Discussed in more detail later, the random distribution of sensor nodes throughout each of the grids allowed isolation of variables through the of number of nodes of each sensor system and that sensor's signal range. Therefore, evaluation of the most efficient application of fusing Wifi, RFID, and GPS to enhance personal navigation in a dense urban environment is feasible.

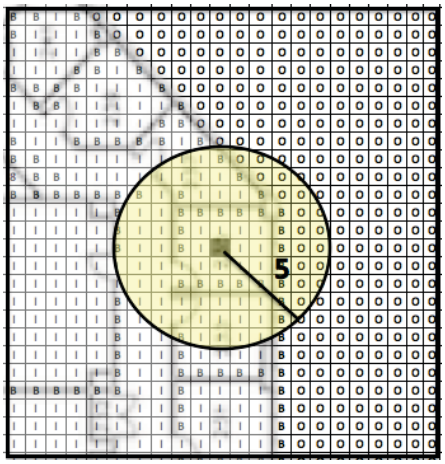


Figure 5.1.: Five cell grid for inside radius

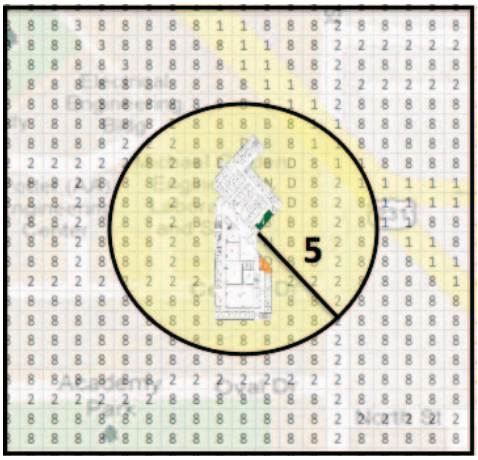


Figure 5.2.: Five cell grid for outside radius

5.1 Methodology

As a recap, the important points to discuss are the building of the agent-based model (ABM) as a discrete-event simulation (DEVS) and the quantification of the data collected.

In building the ABM the characteristics of the model were:

- the model has discrete interactions;
- the model is run by time intervals;
- the agent's position is not fixed;
- the agent population is heterogeneous;
- the topology is modeled to emulate a complex urban environment; and
- the agents exhibit complex behavior.

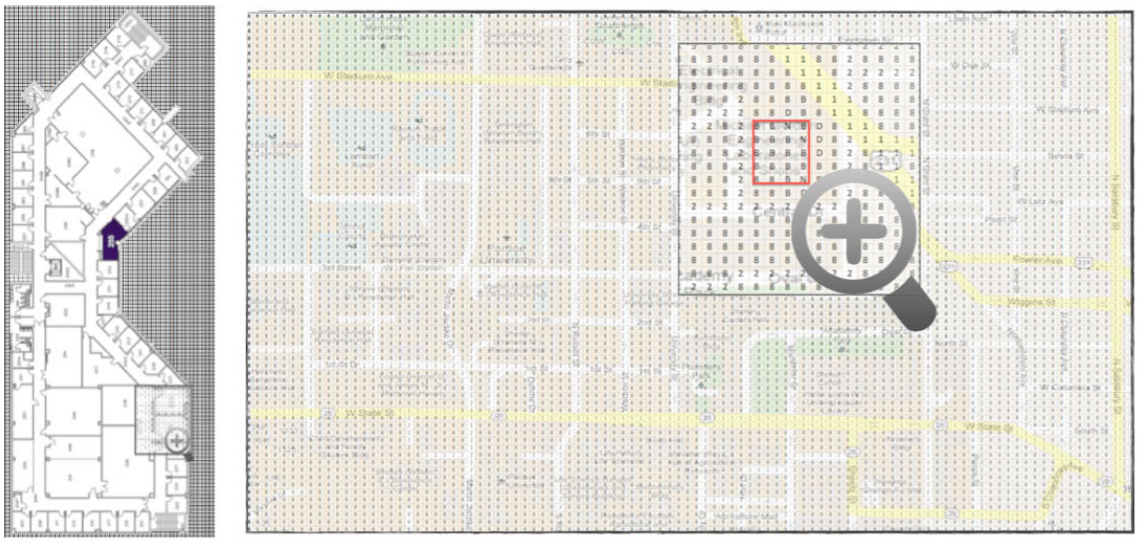


Figure 5.3.: Virtual grid for Purdue Campus & Knoy 2nd Floor

5.2 Design

This model has been designed with the previous iterations in mind. Simplification is required to validate the measurements. As such, the model has been limited to one moving agent. The system is able to function with multiple agents, but to minimize unintended noise, the simulation models one agent per run. Each simulation was launched with the agent leaving room 235, second floor of Maurice G. Knoy Hall of Technology Building, Purdue Campus, 401 North Grant Street, West Lafayette, Indiana, shown in Fig: 5.3. The destination point was room 2142C in The Richard & Patricia Lawson Computer Science Building, Purdue Campus, 305 N. University Street, West Lafayette, Indiana. The path-planning assumed that the agent would leave the Knoy Hall of Technology and proceed to the Grant Street parking garage, drive to the University Street parking garage, and then resume walking to Lawson Computer Science. The full MatLab code is provided in Appendix B. The algorithms are summarized by the functions 2 - 7 on the following pages.

public class main()

Data: plot: [x,y], timestep: $\{t_1, \ldots, t_n\}$, type: clock

begin

for $i = t_1$ *to* t_{n-1} **do**

```
//create adjacency matrices
adjacency matrix = buildMatrix(current[x,y]);
//update path
path, cost = pathPlan(current[x,y], goal[x,y], adjacency matrix);
//update target state and navigation status
state(gps,wifi,rfid) = navState([x,y],t);
```

Path-planning utilized Dijkstra's algorithm to determine the single-source shortest path from the current [x,y] coordinate to either the goal destination or an intermediary goal if the end point is not on the 'local grid'. The cost of the path was returned from the path-planning function to allow for the summation of costs for each trip as a measure of the cost of the run.

The sensor systems maintained an order of precedence depending on its location. When the agent was outside, the agent looked for GPS first, failing that, it checked for Wifi and then proceeded another two time-steps on the known course before checking for a signal again. If the agent was inside, the agent looked for Wifi first, failing that, it checked for RFID, and then proceeded another two time-steps on the known course before checking for either signal again. GPS was assumed unavailable inside; in practice it is imprecise for indoor and elevation information. The class Update Navigation State determines the agent's status (e.g. indoor or outdoor) and then calls the appropriate function. The class returns the updated status of each navigation state.

The GPS update takes the current outdoor position and examines it in relation to the known location of each building. If the current position is within 7 cells of the building, the simulation assigned a probability of loss to calculate if the signal is

public class pathPlan()

```
begin
    //find shortest path
    if (current & goal match (i.e. building & floor)) then
        //find path from current point to destination point
        path, cost = dijkstra(current[x,y], goal[x,y], adjacency matrix);
        return [path, cost];
    else
        intermediate[x,y] = find Nearest Neighbor(current[x,y], closest
        exit);
        //points not compatible must use an intermediate destination
        path, cost = dijkstra(current[x,y], intermediate[x,y], adjacency
        matrix);
        return [path, cost];
```

public class navState([x,y],t)

```
begin
    if (inside) then
        updateWifi([x,y], wifi node locations);
        updateRFID([x,y], rfid node locations);
    else if (outside) then
        updateGPS([x,y], building locations);
        updateWifi([x,y], wifi node locations);
    return updated Nav States;
```

available or not. Once the current user was outside of this range, GPS is assumed to always be available. The state of the GPS signal is returned from this function (GPS update).

The Wifi signal is designed to be updated in the next class 6. A triangle is formed between the agent location and the nearest Wifi node. If the hypotenuse of

public class updateGPS([x,y],t)

begin

//determine the distance from current location to all buildings

for $i = 1$ *to* i_n **do**

distance = $\sqrt{(curX \quad buildingX) + (curY \quad buildingY)^2}$;

//determine if the signal is lost

probability of GPS loss = $\frac{GPSprob(80\%) - GPSprob(80\%)}{7 * min(distance)}$

if *(probability of GPS loss* $\leq$ *0)* **then**

GPS *IS* available;

return GPS status;

else

if *((randomNumber * 100) < probability of GPS loss)* **then**

GPS *IS NOT* available;

return GPS status;

else

GPS *IS* available;

return GPS status;

the triangle is within a given range (e.g. 5, 10 or 15 cells), then the signal is assumed to be available, otherwise it is not. The function returns the state of the Wifi signal to the calling class.

The RFID signal update is similar to the Wifi. In this class 7, a triangle is again formed between the agent location and the nearest RFID node. If the hypotenuse of the triangle is within a given range (e.g. 5, 10 or 15), then the signal is assumed to be available, otherwise it is not. The function returns the state of the RFID signal to the calling class.

The agent is constrained by these simple rules:

1. if the agent is not located in the same building as the goal point, the nearest exit is found;

public class updateWIFI([x,y],t)

begin

//determine the distance from current location to all wifi nodes

for $i = 1$ *to* i_n **do**

distance = $\sqrt{(curX \quad wifiX) + (curY \quad wifiY)^2}$;

//determine if the signal is available

if *(minimum(distance)* $\leq$ *Wifi Range (i.e. 5))* **then**

Wifi *IS* available;

return Wifi status;

else

Wifi *IS NOT* available;

Return Wifi status;

public class updateRFID([x,y],t)

begin

//determine the distance from current location to all rfid nodes

for $i = 1$ *to* i_n **do**

distance = $\sqrt{(curX \quad rfidX) + (curY \quad rfidY)^2}$;

//determine if the signal is available

if *(minimum(distance)* $\leq$ *RFID Range (i.e. 5))* **then**

RFID *IS* available;

return RFID status;

else

RFID *IS NOT* available;

return RFID status;

2. if a graph point contains an obstacle (e.g. a building), the agent must plan a route around it; or

3. if an agent is within a specified range of a building, then with a certain probability P_x, GPS becomes unavailable at that location.

4. if the current signal is unavailable, look for an alternate signal.

5.3 Implementation

Computational Model: M_{IV}

The simulation was implemented in MatLab version 7.11.0.584 (R2010b). The experiment was performed on a MacBook Pro machine configured with a 2.5GHz Quad-core Intel Core i7, 8GB 1333MHz DDR3 SDRAM and a 512GB Solid State Drive.

The map was parsed into coordinates (each cell in the grid was converted into a discrete set of nodes) with adjacency matrices representing pathways which the agents could travel. An extraction of this can be seen in Fig: 5.3 .

Table 5.2: *Table of assumptions for M_{IV}*

Objects	agent	
	objective	target
Variables	position	node[x,y]
	mode of travel	[0=car, 1=walk]
	speed	1:10
	sensor system in use	0=GPS,1=Wifi, 2=RFID, 3=na
	signal availability	[0,1]
	cost	1:10
GPS	probabilistic availability	[60%, 80%]
States	time-step	single step =1
	active sensor system	0=GPS, 1=Wifi, 2=RFID, 3=na
Dynamics	start point	Knoy 235, node[42,107]
	end point	Lawson 2142c, node [41,61]
	Δ travel speed	1:10
Constraints	path does not change	fixed at start
	lowest cost path used	weighted

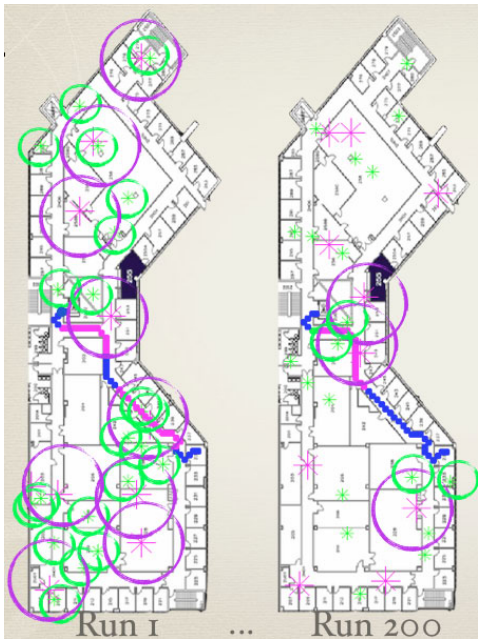


Figure 5.4.: Image of Knoy second floor distributions showing range of sensors

Figure 5.5.: Image of Lawson second floor distributions showing range of sensors

Fig.'s 5.4 and 5.5 show the distribution of sensors on both the first and last simulation. Additionally, the range of each sensor is illustrated. The magenta is for Wifi nodes, green for RFID. The user path is indicated by the line with the same colors, and blue for no signal at all. Fig. 5.6 shows the distribution of Wifi sensors over Purdue campus. The blue outlines indicate the buildings. As the GPS signal was probabilistic within a 7 cell radius, the user's blue line indicates when no GPS signal was available.

The agent modeled dynamic behavior and were able to create real world complexity. Characteristics of the analytical agent-based model are shown in Table: 5.2. This table identifies the variables, states and dynamics and details valid ranges of values for each. The factors are position, mode of travel, speed, sensor system, probabilistic availability, signal availability and cost, and the variables being the changes in GPS probability loss, Wifi and RFID. The position of the agent is given by the

Figure 5.6.: Image of Purdue showing range of sensor distributions

Cartesian coordinate [x,y]. The transportation method is a binary value, 0 for car, 1 for walk. Speed is selected from a range of 1 - 10, and the sensor system in use is designated with numbers: 0 = GPS, 1=Wifi, 2=RFID, 3=no sensor system. Probabilistic availability is a percentage that the GPS is going to fail within 7 cells of a building. The probability is run at both 60% and 80%. Signal availability is a binary value, 0 for none, 1 for signal. The cost function is a minimalization function between 1 and 10, with 1 indicating the lowest cost and 10 the highest.

The states are determined by time-step and active sensor system. These are also defined with a range of values. The time-step is given as a count of the time of each state. The active sensor is a numeric value; 0 for GPS, 1 for Wifi, or 2 for RFID.

Starting point, ending point, and change in travel speed are noted in the system. The starting point for the simulation is fixed in Knoy 235 and the end point

is fixed in Lawson 2142C. A change in travel speed is a range given 1 through 10. This is also a minimalization function with the highest speed being the lowest value. According to a study on the average walking speed [130], humans walk approximately 3.5 miles per hour. The posted speed limit throughout the majority of the Purdue Campus is approximately 25 miles per hour. Converting this to a 1-10 scale with faster speeds being shown as smaller numbers, the scheme is: 1 is driving, 3 is running or biking, 4 is walking outside, 6 and 7 are walking inside, and 10 is slowly moving.

The constraints of the simulation are that the path does not change. Once path-planning is completed, no deviation occurs on the part of the agent (to be addressed in future research), and the lowest cost path is determined and used by a weighted value assigned to each cell of the virtual grid.

Fig: 5.7 show a single run of the simulation when the probability of GPS loss is 80%, range for Wifi is 10 and the range for RFID is 5. The number and distribution of sensors is 0 Wifi and 80 RFID (per floor of each building). The RFID sensors are shown by the black stars and are distributed randomly for each run. The colored line indicates the path of the agent as well as which sensor system is currently active. In these figures, green signals RFID and blue signals that there is no sensor system available.

5.4 Results

In this section, the data is aggregated and analyzed based upon:

- which system is active and providing sensor information;
- the speed allowed by that system;
- the inclusiveness of each sensor system (e.g. indoor and/or outdoor).

These results are then viewed within the context of the hypothesis (introduced in Chapter 1) which permits the testing of the Null Hypothesis and assumptions for the Alternate Hypothesis.

Figure 5.7.: GPS p-loss 80%, 0 Wifi nodes, 80 RFID nodes

- H_0*:* Performing navigation using multiple technologies does not improve at least one of these conditions at any time: availability, speed, or inclusiveness.
- H_a*:* Performing navigation using multiple technologies will improve at least one of these conditions at any time; availability of sensor based on location, speed of navigation, inclusiveness of systems (i.e. outside, inside, elevation, etc.) as the system moves autonomously through individual systems as appropriate. The distributions and measures to evaluate can be seen in Table 5.3.

The table below introduces the distributions to examine (shown in Table: 5.3. This is the same table from Chapter 1, but now with results to examine. The numbers are pulled from included tables (Fig.'s: 5.8, 5.22,5.19), showing the mean of the signal

availability and the speed, as well as the percentage of the availability. Additionally, Fig.'s 5.20 and 5.21 show the percentage increase for the numbers.

Table 5.3: *Distributions to Evaluate Results*

	Navigation of System	Availability of Sensor (0 or 1)	Speed of Navigation (1 - 10 /ts)	Inclusiveness of Systems (Percentage)
1	GPS	{0,1}	52.35	12.68%
	No Signal	{0,0}	257	87.32%
2	GPS	{0,1}	40.67	11.56%
	Wifi	{0,1}	59.39	49.29%
	No Signal	{0,0}	142	34.95%
3	Wifi	{0,1}	59.39	49.29%
	No Signal	{0,0}	142	50.71%
4	Wifi	{0,1}	58.87	49.15%
	RFID	{0,1}	48.05	12.83%
	No Signal	{0,0}	142	38.02%
5	RFID	{0,1}	56.86	14.22%
	No Signal	{0,0}		85.08%
6	GPS	{0,1}	34.68	11.41%
	Wifi	{0,1}	58.87	49.15%
	RFID	{0,1}	48.95	12.83%
	No Signal	{0,0}		26.61%

Wifi was selected at ranges of 5, 10 and 15. This signifies a radius of 5,10 and 15 cells within the virtual checkerboard. These were selected to imitate real-world ranges for Wifi access points. Typical range for indoor access points is 150-300 feet [132], but this is subject to a host of factors besides distance (i.e. interference, congestion, collision avoidance or multi-path signal issues). Interference can be caused by building material or even movable items that are made of thick concrete, metal, glass or aluminum. Florescent lights, microwave ovens, cordless phones and garage door openers have been shown to interfere with wireless signals. Over-crowding of

the base station with users, or overlapping signals causing interference can also cause shortening of the range.

RFID had the highest number of simulation runs with a range of 5. As the project was designed with passive RFID tags in mind, this most resembled an actual scenario. Data was obtained for RFID at ranges of 10 and 15 virtual cells for several scenarios to obtain comparable data to the Wifi statistics.

GPS was modeled with 2 values: probability of signal loss 60% and 80%. These numbers were based on the work of Gartner and Rehrl [133] and their software modeling of an urban canyon.

The number of nodes cycled through 10, 20 and 40, as this provided reasonable coverage throughout the building. Placement of the sensors was not specified. The sensors were placed randomly by the algorithm. As the map is built from a virtual grid, each node is initially assumed acceptable as a sensor location. Once the random selection is made, the software performs a check to be sure that it is in an acceptable location (i.e. no sensor was permitted to be placed in the walls). This is determined by the cost value. Each cell contains a designation between 1 and 10 for a cost, except for those that are unpassable, they are assigned ∞. The nodes are built from the cells of the map; each node represents a cell. Using Matlab notations, the matrix of cells (e.g. [1,1], [1,2], [1,3]... [n,m]) is assigned a node number (e.g. node 1, node 2, node 3... node n). Using the second floor of Knoy as an example, the virtual grid contained a matrix of 51 columns, by 141 rows. Multiplying these returns the number of nodes: 7446.

Approximately 14,000 simulations were run over various configurations. This allowed each unique configuration to be run over 100 times and the data to normalize. The mean cost value of all runs is summarized in Fig: 5.8 in the columns labeled with Wifi, RFID and GPS. The numbers show a sum of the total cost sum for the entire run (357 time-steps). The figure includes the cost of each step (1-10), the cost of the speed (1-10), and a penalty for absence of a signal (2). So for example, line 1 shows a mean for the total cost of each run when Wifi is set to 0 nodes and RFID is also

Total Cost (Step cost (1:10) + Speed (1:10) + Cost Penalty (1:10))

	Range p-loss / Wifi/RFID nodes per floor	Wifi Range 5 Rfid Range 5 GPS loss 60%	Wifi Range 5 Rfid Range 5 GPS loss 80%	Wifi Range 10 Rfid Range 5 GPS loss 60%	Wifi Range 10 Rfid Range 5 GPS loss 80%	Wifi Range 15 Rfid Range 5 GPS loss 60%	Wifi Range 15 Rfid Range 5 GPS loss 80%
1	WIFI 0/RFID 0	18516.84	17894.30	18466.62	18032.18	18284.99	18028.08
2	WIFI 0/RFID 10	9975.82	9391.72	9745.42	9418.68	9820.97	9166.13
3	WIFI 0/RFID 20	6946.66	6809.16	7115.86	6987.82	7057.74	6782.32
4	WIFI 0/RFID 40	5191.44	5110.00	5444.20	4972.52	5238.35	4968.57
5	WIFI 10/RFID 0	9592.24	9048.72	3845.10	3843.32	5844.13	5571.57
6	WIFI 10/RFID 10	6929.86	6764.66	3544.44	3507.62	4834.18	4855.54
7	WIFI 10/RFID 20	5895.08	5646.54	3463.98	3380.20	4320.67	4362.93
8	WIFI 10/RFID 40	4783.08	4474.58	3223.24	3155.18	3812.64	3804.55
9	WIFI 20/RFID 0	6822.04	6674.34	2862.72	2810.16	3855.39	3838.98
10	WIFI 20/RFID 10	5698.90	5498.28	2832.18	2794.34	3688.22	3536.12
11	WIFI 20/RFID 20	4969.10	4802.86	2804.20	2747.38	3453.66	3399.04
12	WIFI 20/RFID 40	4289.46	4158.64	2740.92	2724.44	3231.51	3184.35
13	WIFI 40/RFID 0	4711.84	4707.92	2571.90	2560.32	2894.93	2907.37
14	WIFI 40/RFID 10	4417.18	4257.84	2569.22	2554.70	2884.27	2824.43
15	WIFI 40/RFID 20	4095.42	3984.38	2575.62	2535.82	2832.74	2816.70
16	WIFI 40/RFID 40	3789.54	3720.04	2568.06	2538.86	2791.54	2740.44

Figure 5.8.: Results from simulations with a range of Wifi

set to 0 nodes. The columns across the top indicate the configuration for each run where a variable was altered.

Each run included an agent moving from a fixed starting point to a fixed destination point. Each scenario was run with the probability of GPS signal loss at 60% or 80%. Additionally, the range of the Wifi from the base station was given individual values of 5, 10 or 15. RFID was given values of 5, 10 or 15. Within each of these parameters, the number of randomly placed sensors was modified from 0 Wifi and 0 RFID, to 40 Wifi and 40 RFID. As each run completed, a summary of the total cost was recorded. The cost included the step cost (from the two-dimensional map - line 21), the speed (based on the sensor and location - line 26), and a cost penalty for losing the signal (line 28). These three are tabulated and stored in the Total Cost (line 29). These can be seen in an extraction of agent state over several time-steps as shown in Table: 5.4.

Table 5.4: *A sampling of the state of an agent over a time-step*

	Single Agent Object	t_1	t_2	t_3	t_4	t_5	t_6
1	Start In/Out	1	1	1	1	1	1
2	Start X	42	42	42	42	42	42
3	Start Y	107	107	107	107	107	107
4	Start Floor	2	2	2	2	2	2
5	Start Building	1	1	1	1	1	1
6	Current In/Out	1	1	1	1	1	1
7	Current X	42	41	40	39	38	37
8	Current Y	107	107	107	108	109	108
9	Current Floor	2	2	2	2	2	2
10	Current Building	1	1	1	1	1	1
11	Intermediate In/Out	1	1	1	1	1	1
12	Intermediate X	11	11	11	11	11	11
13	Intermediate Y	75	75	75	75	75	75
14	Intermediate Floor	2	2	2	2	2	2
15	Intermediate Building	1	1	1	1	1	1
16	Goal In/Out	1	1	1	1	1	1
17	Goal X	41	41	41	41	41	41
18	Goal Y	61	61	61	61	61	61
19	Goal Floor	2	2	2	2	2	2
20	Goal Building	2	2	2	2	2	2
21	Step Cost	2	2	2	2	2	2
22	Goal State	0	0	0	0	0	0
23	GPS	0	0	0	0	0	0
24	Wifi	1	1	1	0	0	0
25	RFID	0	0	0	0	0	0
26	Speed	8	8	8	10	10	10
27	Index	0	0	0	0	0	0
28	Penalty	0	0	0	2	4	6
29	Total Step Cost	0	14	14	18	18	20

Fig: 5.9 is designed to show the differences of the data when either the range of the sensor signal is changed, or the number of sensor nodes is changed. The data is plotted to show the changes for Wifi of 10, 20 or 40 or RFID of 10, 20 or 40 when the range of the Wifi signal moves from 5 to 10 to 15. This allowed isolation of each variable to remove noise. The biggest gain is seen when there are 10 Wifi nodes, and the range moves from 5 cells to 10 cells. Wifi and RFID show linear improvement as the number of nodes are increased.

Total Cost (Step cost + Speed + Cost Penalty)

	Wifi/RFID nodes per floor \ Range p-loss	Wifi Range 5 Rfid Range 5 GPS loss 60%	Wifi Range 10 Rfid Range 5 GPS loss 60%	Wifi Range 15 Rfid Range 5 GPS loss 60%
1	WIFI 0/RFID 0	18516.84	18466.62	18284.99
2	WIFI 0/RFID 10	9975.82	9745.42	9820.97
3	WIFI 0/RFID 20	6946.66	7115.86	7057.74
4	WIFI 0/RFID 40	5191.44	5444.20	5238.35
5	WIFI 10/RFID 0	9592.24	3845.10	5844.13
6	WIFI 10/RFID 10	6929.86	3544.44	4834.18
7	WIFI 10/RFID 20	5895.08	3463.98	4320.67
8	WIFI 10/RFID 40	4783.08	3223.24	3812.64
9	WIFI 20/RFID 0	6822.04	2862.72	3855.39
10	WIFI 20/RFID 10	5698.90	2832.18	3688.22
11	WIFI 20/RFID 20	4969.10	2804.20	3453.66
12	WIFI 20/RFID 40	4289.46	2740.92	3231.51
13	WIFI 40/RFID 0	4711.84	2571.90	2894.93
14	WIFI 40/RFID 10	4417.18	2569.22	2884.27
15	WIFI 40/RFID 20	4095.42	2575.62	2832.74
16	WIFI 40/RFID 40	3789.54	2568.06	2791.54

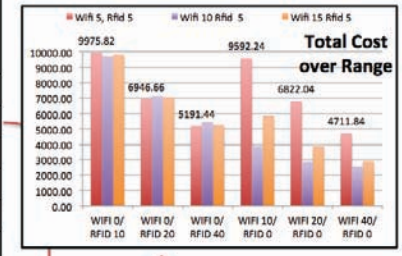


Figure 5.9.: Results from simulations with a range of Wifi and No GPS

Fig: 5.10 shows the affect of the Wifi signal over one-hundred runs of each configuration. RFID was null in these runs to isolate changes in Wifi reception. The blue lines in each graph are when the number of Wifi nodes for each map was set to 10 (e.g. first floor Knoy had 10, second floor Knoy had 10, Purdue map had 10, first floor Lawson had 10, and second floor Lawson had 10). The red line shows the results for 20 Wifi nodes, green shows 40 nodes, and purple shows 80. As expected,

the cost of each simulation improved as the number of Wifi nodes increased. In this graph, the Wifi range was stabilized at 5 cells.

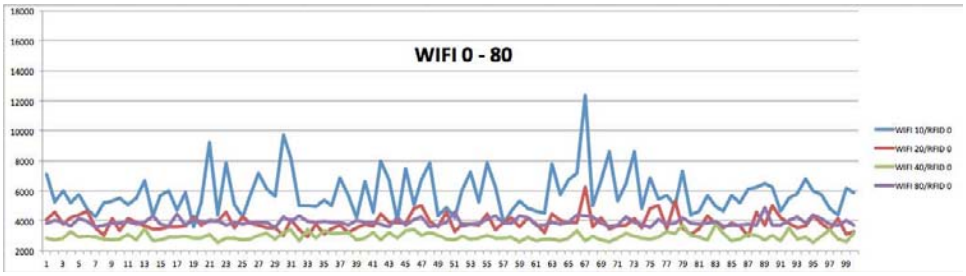


Figure 5.10.: Results from simulations with a range of Wifi

The next graph is designed to show the opposite configuration (Fig: 5.11). In this figure the Wifi nodes have been set to zero, while the RFID count of nodes move between 10, 20, 40 and 80. The RFID sensor range is maintained at a 5 cell radius. Again, there is improvement measured by time-steps of available signal, as the number of sensors increase.

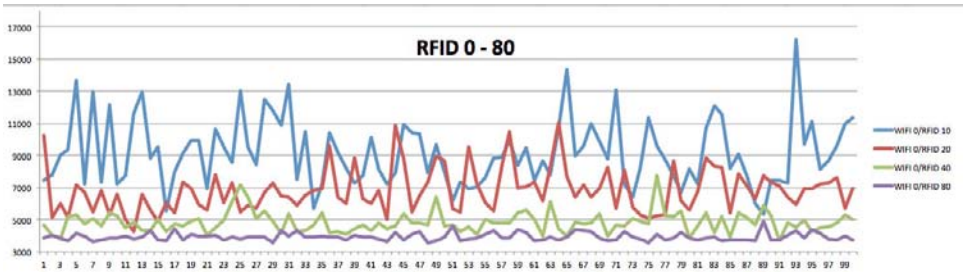


Figure 5.11.: Results from simulations with a range of RFID

The chart in fig: 5.12 compares the Wifi node distribution to the RFID across each sensor count increment (e.g. 10, 20 and 40). In each chart the range of the Wifi and RFID were both 5, while the number of nodes moves between the sensor count equally. The results are approximately equal. To compare, Fig: 5.13 shows identical data, with the exception of the Wifi range, which in this figure, has been increased

from a range of 5 cells to 10 cells. When the number of nodes are equal, Wifi shows an improvement over RFID.

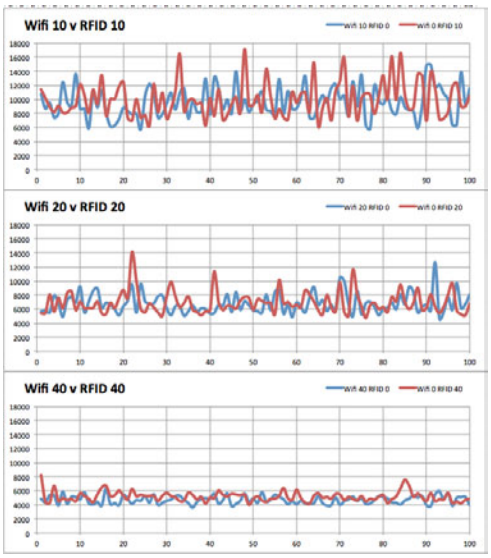


Figure 5.12.: Comparison of Wifi v RFID with equal ranges

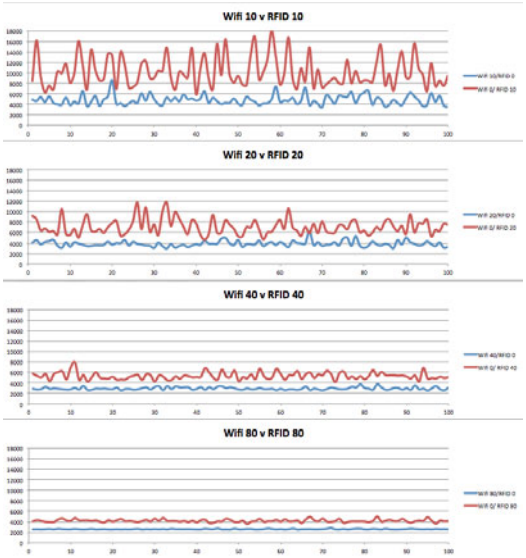


Figure 5.13.: Comparison of Wifi v RFID with ranges of 10 v 5

The experimental setup of the next run compares the difference in values over a changing range of Wifi nodes (Fig: 5.14). The GPS probability of signal loss is held constant at 60%, and the range of RFID sensor signal is 5, but the range of the Wifi sensor signal changes between 5, 10 and 15. In each of the three runs, within each graph, the average over 100 runs show improvement for each iteration as the number of each sensor is increased. The previous result is maintained but it is important to note that the change in Wifi range from 5 cells to 10 cells shows the biggest performance increase.

Fig: 5.15 shows the same Wifi configuration with RFID removed. The GPS probability of signal loss is constant at 60% and the Wifi range moves between 5, 10 and 15. For each run, an interval graph is shown with the improvement for both number of Wifi nodes and the range.

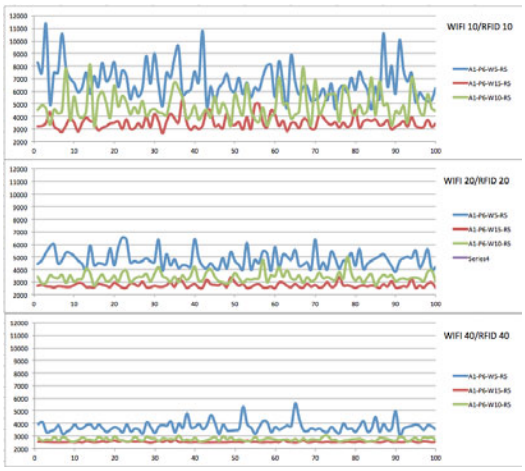


Figure 5.14.: Results for the difference in the Wifi range with 10-40 Wifi and 10-40 RFID

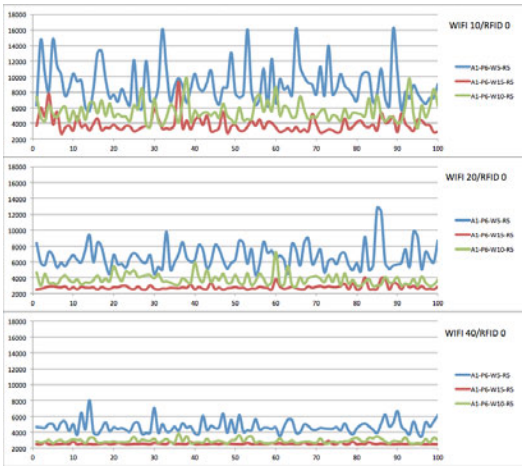


Figure 5.15.: Results for the difference in the Wifi range with 0-40 Wifi and 0 RFID nodes

The same result can be seen in the next figure (Fig: 5.16), which emphasizes this point by showing each Wifi distribution (e.g. 0,10, 20 and 40) for increasing Wifi ranges (e.g. 5,10 and 15). The number of nodes as well as the range of each sensors impact performance, but the improvement from the difference of range is greater.

The difference in RFID is illustrated in Fig's: 5.17 and 5.18. Each chart plots the difference when RFID moves from 10 to 20 and 40, and between ranges 5, 10 and 15. There is an improvement based on the number of nodes as well as the range of the signal, however the Fig: 5.17 shows more pronounced improvement (signal range) over Fig: 5.18 (all nodes).

The next figure illustrates the absolute number of signal availability over each run (Fig: 5.19). As the simulation progresses by time-step, the state of the agent is logged including the measure of signal availability. During the run, time-steps 1-73 occurred within Knoy Hall of Technology. Time-steps 74 - 210 occurred outside on the Purdue campus and time-steps 211 - 357 were logged inside Lawson Computer

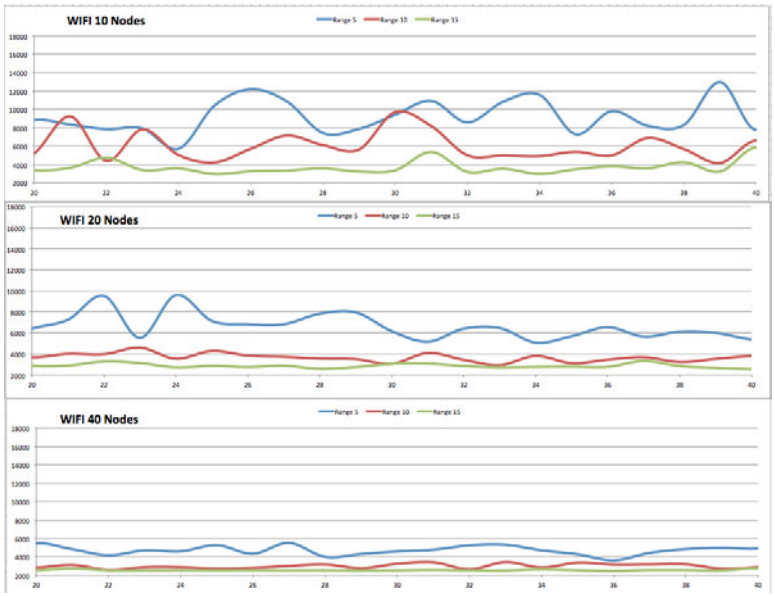


Figure 5.16.: Results of Wifi distributions over different signal ranges

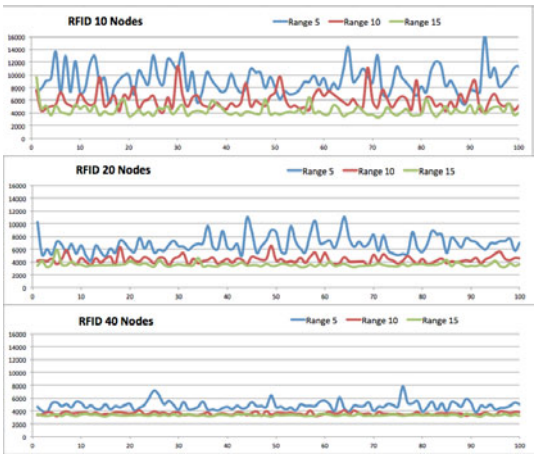


Figure 5.17.: Distribution of RFID across all nodes

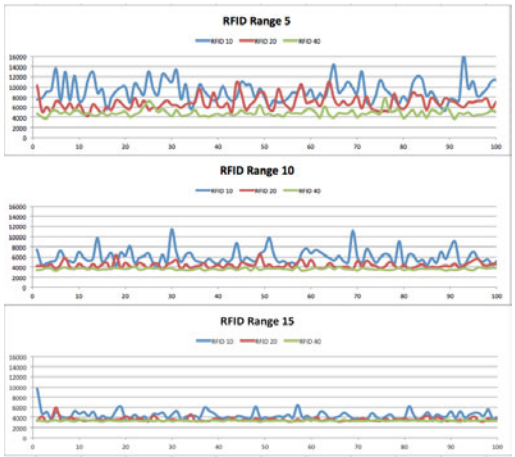


Figure 5.18.: Distribution of RFID across all ranges

Science. GPS was only available during the outside portion, RFID during the inside portion and Wifi across the entire trip.

For this reason, the summary of signals (1 indicates a signal, 0 for no signal) is shown only for the time-steps they were available; out of 136 time-steps for GPS; 357 time-steps for Wifi; and 221 time-steps for RFID. The table shows the mean of each 100 runs for each configuration: GPS loss probability 60% and Wifi range 5 or 10 or 15; and GPS loss probability of 80% and Wifi range 5 or 10 or 15. RFID is stable at a range of 5 cells for these runs. Within each distribution above, the count of Wifi and RFID nodes (per floor of each building) move through increments of 0, 10, 20 and 40.

Signal Mean

Range / p-loss / Wifi/RFID nodes per floor	GPS loss p-60% 136/357	WIFI range 5 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-60% 136/357	WIFI range 10 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-60% 136/357	WIFI range 15 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-80% 136/357	WIFI range 5 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-80% 136/357	WIFI range 10 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-80% 136/357	WIFI range 15 cells Out of 357	RFID range 5 cells 218/357
0-0	90.56	0	0	85.88	0	0	88	0	0	67.92	0	0	67.84	0	0	70.92	0	0
0-10	90.96	0	36.46	82.26	0	28.53	83.34	0	28.64	75.32	0	30.81	70.08	0	27.26	70.82	0	27.9
0-20	92.22	0	48	83.93	0	61.42	83.1	0	54.55	67.46	0	56.16	80.62	0	55.24	68.64	0	52.64
0-40	92.22	0	94.84	84.62	0	101.56	80.8	0	94.7	67.04	0	101.76	66.36	0	103.96	65	0	90.85
10-0	80.22	48.54	0	81.24	147.74	0	82.16	246.66	0	70.36	46.36	0	65.16	146.66	0	66.24	244.38	0
10-10	91.84	49.98	28.49	84.24	151.38	25.93	88.62	242.6	25.83	75.38	49.08	25.96	73.08	151.4	30.65	67.4	247.96	32.12
10-20	84.26	45.04	52.33	84.32	146.94	56.02	84.52	239.5	59.06	67.12	45.62	56.4	65.96	144.28	55.67	71.96	245.4	52.88
10-40	78	45.48	97.22	84.44	154.62	90.65	85.74	246.34	99.32	62.42	44.9	97.16	65.3	147.3	97.09	68.64	245.72	93.95
20-0	83.92	88.6	0	85.26	232.46	0	89.2	319.28	0	65.14	86.26	0	71.74	226.9	0	69.08	319.4	0
20-10	82.22	84	25.96	88.92	232.78	30.01	85.66	319.88	31.44	62.54	87.48	29.22	69.32	224.28	36.54	63.86	318.62	24.28
20-20	89.42	85.04	60.16	79.16	231.88	56.99	85.66	319.88	31.44	90.2	319.78	53.13	73.38	229	42.31	66.26	318.22	58.23
20-40	88.78	84.56	96.76	76.28	232.1	96.24	84.74	316.5	97.92	57.96	86.52	100.8	68.78	228.32	96.9	67.34	318.2	95.2
40-0	88.94	151.34	0	83.2	306.22	0	88.36	349.92	0	67.98	154.7	0	71.98	305.28	0	71	351.04	0
40-10	81.8	149.36	28.32	79.8	308.04	29.49	85.86	349.78	28.81	64.86	151.58	28.36	66.52	306.6	27.68	79.78	350.72	30.08
40-20	83.68	148.66	57.9	86.8	307.4	49.97	82.56	351.92	55.87	72.8	153.48	61.69	65.9	307.82	57.48	71.38	350.16	50.02
40-40	83.22	151.74	95.71	85.44	305.74	99.92	81.46	350.9	91.64	67.66	149.06	102.91	63.74	306.06	98.2	69.02	350.5	92.25

Figure 5.19.: A table of the mean value for each 100 runs

Fig.'s: 5.22 and 5.23 were configured to show the percent of availability of each signal over the time-steps. GPS was potentially available for 136 time-steps (the number of time-steps that occurred outside), RFID for 221 time-steps (the number of time-steps from inside the buildings) and Wifi for 357 (the time-step of the total run). The mean of both GPS and RFID were divided by 357 and the mean of Wifi divided by 714 and formatted as a percentage. The percentage of time with no signal was 1 minus the sum of the three percentages.

Signal Availability Percentage

Range p-loss / Wifi/RFID nodes per floor	GPS loss p-60% 136/357	WIFI range 5 cells Out of 357	RFID range 5 cells 218/357	No Signal Out of 357	GPS loss p-60% 136/	Wifi range 10 cells Out of	RFID range 5 cells 218/	No Signal Out of 357	GPS loss p-60% 136/357	Wifi range 15 cells Out of 357	RFID range 5 cells 218/357	No Signal Out of 357
0-0	25.37%	0.00%	0.00%	74.63%					24.65%	0.00%	0.00%	75.35%
0-10	25.48%	0.00%	10.21%	64.31%					23.34%	0.00%	8.02%	68.63%
0-20	25.83%	0.00%	13.45%	60.72%					23.28%	0.00%	15.28%	61.44%
0-40	25.83%	0.00%	26.57%	47.60%					22.63%	0.00%	26.53%	50.84%
10-0	22.47%	6.80%	0.00%	70.73%					23.01%	34.55%	0.00%	42.44%
10-10	25.73%	7.00%	7.98%	59.29%					24.82%	33.98%	7.24%	33.96%
10-20	23.60%	6.31%	14.66%	55.43%					23.68%	33.54%	16.54%	26.24%
10-40	21.85%	6.37%	27.23%	44.55%					24.02%	34.50%	27.82%	13.66%
20-0	23.51%	12.41%	0.00%	64.08%					24.99%	44.72%	0.00%	30.30%
20-10	23.03%	11.76%	7.27%	57.93%					23.99%	44.80%	8.81%	22.40%
20-20	25.05%	11.91%	16.85%	46.19%					23.99%	44.80%	8.81%	22.40%
20-40	24.87%	11.84%	27.10%	36.18%					23.74%	44.33%	27.43%	4.51%
40-0	24.91%	21.20%	0.00%	53.89%					24.75%	49.01%	0.00%	26.24%
40-10	22.91%	20.92%	7.93%	48.24%					24.05%	48.99%	8.07%	18.89%
40-20	23.44%	20.82%	16.22%	39.52%	24.31%	43.05%	14.00%	18.64%	23.13%	49.29%	15.65%	11.94%
40-40	23.31%	21.25%	26.81%	28.63%	23.93%	42.82%	27.99%	5.26%	22.82%	49.15%	25.67%	2.37%

	RANGE 5
RFID 10	29.12
RFID 20	53.98
RFID 40	96.98
% INC 10-20	85.41%
% INC 20-40	79.65%

Figure 5.20.: A table of the percentage increase for RFID

Fig: 5.24 shows the standard deviation among each set of runs (100 - 300). Although there is fluctuation, the running of the simulations repeatedly allows the data to normalize.

5.4.1 Statistical Analysis

Each of the simulations was distributed across patterns of configurations: Wifi sensor range of 5 or 10 or 15 cells with the count of RFID sensors at 0; and then RFID sensor range of 5 or 10 or 15 cells with the number of Wifi nodes set to zero. Within these 6 categories, the simulations were run with 10, 20 and 40 sensor nodes for each category. The simulation data is examined using histograms and Pearson's Chi-

Signal Availability Percentage

Range p-loss / Wifi/RFID nodes per floor	GPS loss p-60%	WIFI range 5 cells	RFID range 5 cells	No Signal
	136/357	Out of 357	218/357	Out of 357
0-0	25.37%	0.00%	0.00%	74.63%
0-10	25.48%	0.00%	10.21%	64.31%
0-20	25.83%	0.00%	13.45%	60.72%
0-40	25.83%	0.00%	26.57%	47.60%
10-0	22.47%	6.80%	0.00%	70.73%
10-10	25.73%	7.00%	7.98%	59.29%
10-20	23.60%	6.31%	14.66%	55.43%
10-40	21.85%	6.37%	27.23%	44.55%
20-0	23.51%	12.41%	0.00%	64.08%
20-10	23.03%	11.76%	7.27%	57.93%
20-20	25.05%	11.91%	16.85%	46.19%
20-40	24.87%	11.84%	27.10%	36.18%
40-0	24.91%	21.20%	0.00%	53.89%
40-10	22.91%	20.92%	7.93%	48.24%
40-20	23.44%	20.82%	16.22%	39.52%
40-40	23.31%	21.25%	26.81%	28.63%

	GPS loss p-60%	Wifi range 10 cells	RFID range 5 cells	No Signal
	136/357	Out of 357	218/357	Out of 357
	24.31%	43.05%	14.00%	18.64%
	23.93%	42.82%	27.99%	5.26%

	GPS loss p-60%	Wifi range 15 cells	RFID range 5 cells	No Signal
	136/357	Out of 357	218/357	Out of 357
	23.13%	49.29%	15.65%	11.94%
	22.82%	49.15%	25.67%	2.37%

	RANGE 5	RANGE 10	RANGE 15	% INC 5-10	% INC 10-15
WIFI 10	46.88	148.8	244.85	217.42%	64.56%
WIFI 20	115.28	229.7	318.75	114.09%	38.78%
WIFI 40	151.25	306.65	350.6	102.67%	14.34%
% INC 10-20	81.02%	54.69%	30.81%		
% INC 20-40	75.66%	32.09%	9.95%		

Figure 5.21.: A table of the percentage increase for Wifi

Square Test of Independence. This is designed to characterize, model and analyze the data to observe behavior.

The final comparison is Pearson's Chi-Squared Test of Independence for the Wifi distribution. Pearson's Chi-Squared Test is defined by:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i \quad E_i)^2}{E_i} \tag{5.1}$$

where O_i is the observed frequency, and E_i is the expected frequency [134]. Wifi is run for the range 5, 10 and 15 with the number of nodes set to 10, 20 and 40. RFID is also run for a range of 5, 10 and 15 cells with the number of nodes equal to 10, 20 and 40 randomly distributed throughout each building. The scores are the time-steps that there is a usable signal. The following hypothesis is tested:

Signal Availability Percentage

Range p-loss / Wifi/RFID nodes per floor	GPS loss p-60%	WIFI range 5 cells	RFID range 5 cells	No Signal	GPS loss p-60%	Wifi range 10 cells	RFID range 5 cells	No Signal	GPS loss p-60%	Wifi range 15 cells	RFID range 5 cells	No Signal
	136/ 357	Out of 357	218/ 357	Out of 357	136/ 357	Out of 357	218/ 357	Out of 357	136/ 357	Out of 357	218/ 357	Out of 357
0-0	25.37%	0.00%	0.00%	74.63%	24.06%	0.00%	0.00%	75.94%	24.65%	0.00%	0.00%	75.35%
0-10	25.48%	0.00%	10.21%	64.31%	23.04%	0.00%	7.99%	68.97%	23.34%	0.00%	8.02%	68.63%
0-20	25.83%	0.00%	13.45%	60.72%	23.51%	0.00%	17.20%	59.29%	23.28%	0.00%	15.28%	61.44%
0-40	25.83%	0.00%	26.57%	47.60%	23.70%	0.00%	28.45%	47.85%	22.63%	0.00%	26.53%	50.84%
10-0	22.47%	6.80%	0.00%	70.73%	22.76%	20.69%	0.00%	56.55%	23.01%	34.55%	0.00%	42.44%
10-10	25.73%	7.00%	7.98%	59.29%	23.60%	21.20%	7.26%	47.94%	24.82%	33.98%	7.24%	33.96%
10-20	23.60%	6.31%	14.66%	55.43%	23.62%	20.58%	15.69%	40.11%	23.68%	33.54%	16.54%	26.24%
10-40	21.85%	6.37%	27.23%	44.55%	23.65%	21.66%	25.39%	29.30%	24.02%	34.50%	27.82%	13.66%
20-0	23.51%	12.41%	0.00%	64.08%	23.88%	32.56%	0.00%	43.56%	24.99%	44.72%	0.00%	30.30%
20-10	23.03%	11.76%	7.27%	57.93%	24.91%	32.60%	8.41%	34.08%	23.99%	44.80%	8.81%	22.40%
20-20	25.05%	11.91%	16.85%	46.19%	22.17%	32.48%	15.96%	29.39%	23.99%	44.80%	8.81%	22.40%
20-40	24.87%	11.84%	27.10%	36.18%	21.37%	32.51%	26.96%	19.17%	23.74%	44.33%	27.43%	4.51%
40-0	24.91%	21.20%	0.00%	53.89%	23.31%	42.89%	0.00%	33.81%	24.75%	49.01%	0.00%	26.24%
40-10	22.91%	20.92%	7.93%	48.24%	22.35%	43.14%	8.26%	26.24%	24.05%	48.99%	8.07%	18.89%
40-20	23.44%	20.82%	16.22%	39.52%	24.31%	43.05%	14.00%	18.64%	23.13%	49.29%	15.65%	11.94%
40-40	23.31%	21.25%	26.81%	28.63%	23.93%	42.82%	27.99%	5.26%	22.82%	49.15%	25.67%	2.37%

Figure 5.22.: Percentage of signal availability over 100 runs at p-loss 60%

H_{0_1} = increasing the number of Wifi nodes does not cause an increased performance for each range that is tested.

H_{A_1} = increasing the number of Wifi nodes does cause an increased performance for each range that is tested.

The $p \quad value$ for Wifi returns 0.00009%. This is far less than the accepted statistical value of significance of 0.05% or even 0.01%. Therefore, it can be concluded that the null hypothesis (H_{0_1}) is false , and the alternate hypothesis (H_{A_1}) is accepted.

Pearson's Chi-Squared Test of Independence was also run for the RFID distribution. Pertaining to RFID, the hypothesis tested is:

H_{0_2} = increasing the number of RFID nodes does not cause an increased performance for each range that is tested.

Signal Availability Percentage

Range p-loss / Wifi/RFID nodes per floor	GPS loss p-80%	WIFI range 5 cells	RFID range 5 cells	No Signal	GPS loss p-80%	Wifi range 10 cells	RFID range 5 cells	No Signal	GPS loss p-80%	Wifi range 15 cells	RFID range 5 cells	No signal
	136/ 357	Out of 357	218/ 357	Out of 357	136/ 357	Out of 357	218/ 357	Out of 357	136/ 357	Out of 357	218/ 357	Out of 357
0-0	19.03%	0.00%	0.00%	80.97%	19.00%	0.00%	0.00%	81.00%	19.87%	0.00%	0.00%	80.13%
0-10	21.10%	0.00%	8.63%	70.27%	19.63%	0.00%	7.64%	72.73%	19.84%	0.00%	7.82%	72.35%
0-20	18.90%	0.00%	15.73%	65.37%	22.58%	0.00%	15.47%	61.94%	19.23%	0.00%	14.75%	66.03%
0-40	18.78%	0.00%	28.50%	52.72%	18.59%	0.00%	29.12%	52.29%	18.21%	0.00%	25.45%	56.34%
10-0	19.71%	6.49%	0.00%	73.80%	18.25%	20.54%	0.00%	61.21%	18.55%	34.23%	0.00%	47.22%
10-10	21.11%	6.87%	7.27%	64.74%	20.47%	21.20%	8.59%	49.74%	18.88%	34.73%	9.00%	37.39%
10-20	18.80%	6.39%	15.80%	59.01%	18.48%	20.21%	15.59%	45.72%	20.16%	34.37%	14.81%	30.66%
10-40	17.48%	6.29%	27.22%	49.01%	18.29%	20.63%	27.20%	33.88%	19.23%	34.41%	26.32%	20.04%
20-0	18.25%	12.08%	0.00%	69.67%	20.10%	31.78%	0.00%	48.13%	19.35%	44.73%	0.00%	35.92%
20-10	17.52%	12.25%	8.18%	62.04%	19.42%	31.41%	10.24%	38.94%	17.89%	44.62%	6.80%	30.69%
20-20	25.27%	44.79%	14.88%	15.06%	20.55%	32.07%	11.85%	35.52%	18.56%	44.57%	16.31%	20.56%
20-40	16.24%	12.12%	28.24%	43.41%	19.27%	31.98%	27.14%	21.61%	18.86%	44.57%	26.67%	9.90%
40-0	19.04%	21.67%	0.00%	59.29%	20.16%	42.76%	0.00%	37.08%	19.89%	49.17%	0.00%	30.95%
40-10	18.17%	21.23%	7.94%	52.66%	18.63%	42.94%	7.75%	30.67%	22.35%	49.12%	8.43%	20.11%
40-20	20.39%	21.50%	17.28%	40.83%	18.46%	43.11%	16.10%	22.33%	19.99%	49.04%	14.01%	16.95%
40-40	18.95%	20.88%	28.83%	31.34%	17.85%	42.87%	27.51%	11.77%	19.33%	49.09%	25.84%	5.74%

Figure 5.23.: Percentage of signal availability over 100 runs at p-loss 80%

H_{A_2} = increasing the number of RFID nodes does cause an increased performance for each range that is tested.

The p *value* for RFID returns 0.002%. This is also below the fields required statistical value of 0.05% or 0.01%, so it can be concluded that this null hypothesis (H_{0_2}) is false, and this alternate hypothesis (H_{A_2}) is accepted.

Histograms are presented in Fig:'s 5.27, 5.28 and 5.29. The histogram is a visual depiction of the distribution of the data [135]. Relative frequency histograms are shown for Wifi and RFID, with the same cell ranges (e.g. 5, 10 and 15) and variable nodes (e.g. 10, 20, 40) in Fig.'s: 5.27 - 5.28. Additionally, GPS was compared for the values 60% and 80% probability of signal loss in Fig's: 5.29. The minimum, maximum, mean and standard deviation are shown for each category. The categories

Std. Dev. Of Mean Signal

Range p-loss / Wifi/RFID nodes per floor	GPS loss p-60% 136/357	WIFI range 5 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-60% 136/357	WIFI range 10 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-60% 136/357	WIFI range 15 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-80% 136/357	WIFI range 5 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-80% 136/357	WIFI range 10 cells Out of 357	RFID range 5 cells 218/357	GPS loss p-80% 136/357	WIFI range 15 cells Out of 357	RFID range 5 cells 218/357
0-0	36.045	0	0	37.24	0	0	37.211	0	0	38.53	0	0	38.557	0	0	37.708	0	0
0-10	35.557	0	33.225	38.335	0	23.574	37.738	0	25.984	38.472	0	28.973	38.33	0	22.968	38.362	0	24.432
0-20	35.2	0	26.148	37.892	0	51.287	37.639	0	33.47	37.673	0	31.642	36.97	0	35.712	38.584	0	32.806
0-40	36.263	0	39.161	37.367	0	39.252	38.418	0	40.479	37.192	0	38.697	38.195	0	42.142	38.169	0	37.486
10-0	38.937	19.185	0	37.814	30.873	0	38.838	35.258	0	38.147	17.25	0	38.48	31.215	0	37.99	34.293	0
10-10	36.185	17.533	22.16	37.705	30.771	24.815	36.704	32.738	19.587	39.023	14.611	22.074	38.267	27.81	25.887	38.12	31.438	29.14
10-20	37.481	17.565	31.544	36.999	31.843	35.377	37.75	34.285	37.959	38.243	16.622	35.672	37.859	31.291	34.47	38.386	35.447	28.465
10-40	39.103	16.423	41.349	37.63	33.697	36.135	36.71	31.948	38.544	37.151	13.952	40.065	38.469	32.597	39.612	38.63	31.702	37.437
20-0	38.069	22.913	0	37.705	32.627	0	37.419	20.055	0	37.811	19.758	0	38.563	27.965	0	38.478	21.235	0
20-10	37.638	21.53	16.072	36.976	30.723	22.111	38.238	21.839	26.755	37.334	21.393	23.534	38.179	30.324	33.206	37.516	19.197	18.487
20-20	36.278	20.466	34.623	37.72	26.298	36.602	38.238	21.839	26.755	36.364	20.145	33.464	38.124	31.294	22.8	37.073	21.592	33.294
20-40	35.918	22.244	40.295	38.17	30.926	41.576	38.143	23.635	39.224	35.223	21.322	39.534	37.969	31.256	40.753	38.494	20.625	40.52
40-0	36.586	23.05	0	37.876	20.457	0	36.614	7.6989	0	38.263	23.1	0	39.133	19.792	0	38.625	6.6602	0
40-10	38.601	23.253	23.193	38.393	19.702	27.166	37.931	7.5572	24.04	36.852	26.94	24.759	38.161	22.616	25.83	38.891	7.5764	25.897
40-20	37.435	25.152	34.99	37.101	21.239	29.579	37.944	5.6915	34.17	38.475	22.494	36.633	38.047	19.206	34.486	38.187	7.6088	30.684
40-40	38.444	24.288	38.843	37.686	19.681	39.482	38.072	7.2794	36.546	37.885	22.311	41.426	37.769	21.977	41.8	38.75	7.4047	39.096

Figure 5.24.: The standard deviation of the mean

are made up of at least 100 runs each. The frequencies are spread into 'bins' and then the distribution is both listed and shown in a chart. For the Wifi data, there is discernible movement both for the increase in sensor nodes (shown in each chart), and the range of the sensors (shown vertically in each chart). Most of the histograms for Wifi approach a normal distribution with the exception of a few outliers.

The histograms created for the RFID distributions also show the minimum, maximum, mean and standard deviation. The relative frequency of each 100 runs is broken into a separate information section. Although RFID shows an improvement for an increase in the number of sensors, it does not show the same increase for the higher cell range for the sensor signal. The data shows a noticeable dip from the RFID range of 10 cells, to the higher range of 15 cells. Additionally, the data appears more sinusoidal than a normalized bell-curve.

The histogram created for the GPS distribution examines the relationship between the random probability of signal loss for both 60% and 80%. There is no range for the GPS signal. It is considered full-coverage outside, unless the proximity of a building interferes. There is no consideration for nodes in this context either. As GPS is predicated on satellites, there is little chance of adding or removing satellites

Empirical Data Values

Wifi Mean	10 nodes	20 nodes	40 nodes	Node Total
Range 5	48.54	88.6	151.34	288.48
Range 10	147.74	230.46	306.22	684.42
Range 15	246.66	319.28	349.92	915.86
Range Total	442.94	638.34	807.48	1888.76

Exected Data Values

Wifi Mean	10 nodes	20 nodes	40 nodes	Node Total
Range 5	67.65	97.50	123.33	288.48
Range 10	160.51	231.31	292.60	684.42
Range 15	214.78	309.53	391.55	915.86
Range Total	442.94	638.34	807.48	1888.76

Chi-Square Test of Independence

p-value	9.22E-05
° of Freedom	4

Figure 5.25.: Chi-Square Test of Independence for Wifi

from the equation for any type of environment. With the previous limitations, the data from multiple runs is incorporated into Fig: 5.29. While the previous histograms cover approximately 100 runs, GPS includes 1,000 runs for each coverage level. The data exhibits an improvement when the probability of GPS loss is lower, but also displays a bimodal distribution with no data in between the distributions.

5.5 Conclusion

This chapter presented the final model and the data captured during the runs which the model simulated. Over 14,000 iterations were run resulting in identifiable but modest changes in the three critical independent variables. These changes were associated with (a) increase in the number of Wifi and RFID sensors, (b) increase in

Empirical Data Values

RFID mean	10 nodes	20 nodes	40 nodes	Node Total
Range 5	36.46	48	94.84	179.3
Range 10	94.6	141.56	195.61	431.77
Range 15	159.19	141.57	215.63	516.39
Range Total	290.25	331.13	506.08	1127.46

Expected Data Values

RFID Mean	10 nodes	20 nodes	40 nodes	Node Total
Range 5	46.16	52.66	80.48	179.3
Range 10	111.15	126.81	193.81	431.77
Range 15	132.94	151.66	231.79	516.39
Range Total	290.25	331.13	506.08	1127.46

Chi-Square Test of Independence

p-value	0.0027722
° of Freedom	4

Figure 5.26.: Chi-Square Test of Independence for RFID

range of sensors and (c) combination sensor signals into single application capable of exploiting the incoming information via the prototype software.

An image of the simulation in action is given with Fig:'s 5.30, 5.31, 5.32 and 5.33. In the image each of the Wifi and RFID sensors location is shown, Wifi is magenta, while RFID is black. The path of the user is shown based on the signal in use. The legend is included (Fig: 5.31), with Wifi as magenta, RFID as green, GPS as black and blue when there is no signal at all.

The data indicated that for a typical mid-range public building, the best improvement in signal availability (i.e. higher time-step count) was a mid Wifi range of 10 cells and a sensor count of 20 nodes and an RFID range of 5 cells and a sensor count of 40 nodes. The nodes are not required to be distributed in a grid fashion,

Descriptive statistics (WIFI 10):

Min:	Max:	Mean:	St Dev:
8	106	46	17

Lower bound	Upper bound	Freq	Relative freq	Density
0	11	1	0.01	0.00
11	22	3	0.03	0.00
22	33	21	0.21	0.02
33	44	20	0.20	0.02
44	55	25	0.25	0.02
55	66	16	0.16	0.01
66	77	8	0.08	0.01
77	88	5	0.05	0.00
88	99	0	0.00	0.00
99	110	1	0.01	0.00

Descriptive statistics (WIFI 20):

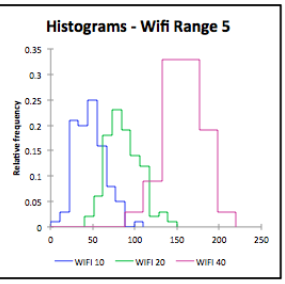
Min:	Max:	Mean:	St Dev:
48	140	86	20

Lower bound	Upper bound	Freq	Relative freq	Density
40	51	2	0.02	0.00
51	62	6	0.06	0.01
62	73	18	0.18	0.02
73	84	23	0.23	0.02
84	95	19	0.19	0.02
95	106	14	0.14	0.01
106	117	12	0.12	0.01
117	128	2	0.02	0.00
128	139	3	0.03	0.00
139	150	1	0.01	0.00

Descriptive statistics (WIFI 40):

Min:	Max:	Mean:	St Dev:
92	206	155	23

Lower bound	Upper bound	Freq	Relative freq	Density
0	22	0	0.00	0.00
22	44	0	0.00	0.00
44	66	0	0.00	0.00
66	88	0	0.00	0.00
88	110	3	0.03	0.00
110	132	9	0.09	0.00
132	154	33	0.33	0.02
154	176	33	0.33	0.02
176	198	19	0.19	0.01
198	220	3	0.03	0.00



Descriptive statistics (WIFI 10):

Min:	Max:	Mean:	St Dev:
72	222	147	31

Lower bound	Upper bound	Freq	Relative freq	Density
0	24	0	0.00	0.00
24	48	0	0.00	0.00
48	72	0	0.00	0.00
72	96	3	0.03	0.00
96	120	19	0.19	0.01
120	144	25	0.25	0.01
144	168	28	0.28	0.01
168	192	17	0.17	0.01
192	216	6	0.06	0.00
216	240	2	0.02	0.00

Descriptive statistics (WIFI 20):

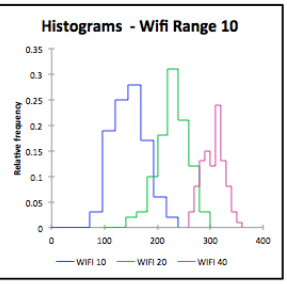
Min:	Max:	Mean:	St Dev:
156	284	227	28

Lower bound	Upper bound	Freq	Relative freq	Density
100	120	0	0.00	0.00
120	140	0	0.00	0.00
140	160	2	0.02	0.00
160	180	3	0.03	0.00
180	200	10	0.10	0.01
200	220	18	0.18	0.01
220	240	31	0.31	0.02
240	260	21	0.21	0.01
260	280	12	0.12	0.01
280	300	3	0.03	0.00

Descriptive statistics (WIFI 40):

Min:	Max:	Mean:	St Dev:
262	350	305	20

Lower bound	Upper bound	Freq	Relative freq	Density
260	270	3	0.03	0.00
270	280	8	0.08	0.01
280	290	13	0.13	0.01
290	300	15	0.15	0.02
300	310	12	0.12	0.01
310	320	24	0.24	0.02
320	330	13	0.13	0.01
330	340	8	0.08	0.01
340	350	3	0.03	0.00
350	360	1	0.01	0.00



Descriptive statistics (WIFI 10):

Min:	Max:	Mean:	St Dev:
156	314	244	34

Lower bound	Upper bound	Freq	Relative freq	Density
100	123	0	0.00	0.00
123	146	0	0.00	0.00
146	169	1	0.01	0.00
169	192	3	0.03	0.00
192	215	19	0.19	0.01
215	238	21	0.21	0.01
238	261	21	0.21	0.01
261	284	19	0.19	0.01
284	307	13	0.13	0.01
307	330	3	0.03	0.00

Descriptive statistics (WIFI 20):

Min:	Max:	Mean:	St Dev:
246	354	319	21

Lower bound	Upper bound	Freq	Relative freq	Density
200	217	0	0.00	0.00
217	234	0	0.00	0.00
234	251	1	0.01	0.00
251	268	2	0.02	0.00
268	285	4	0.04	0.00
285	302	12	0.12	0.01
302	319	25	0.25	0.01
319	336	29	0.29	0.02
336	353	26	0.26	0.02
353	370	1	0.01	0.00

Descriptive statistics (WIFI 40):

Min:	Max:	Mean:	St Dev:
326	356	351	7

Lower bound	Upper bound	Freq	Relative freq	Density
320	324	0	0.00	0.00
324	328	2	0.02	0.01
328	332	0	0.00	0.00
332	336	1	0.01	0.00
336	340	4	0.04	0.01
340	344	8	0.08	0.02
344	348	2	0.02	0.01
348	352	19	0.19	0.05
352	356	20	0.20	0.05
356	360	44	0.44	0.11

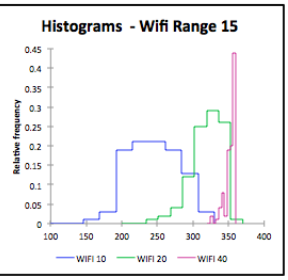


Figure 5.27.: Histogram of all Wifi configurations

placement is secondary to the count and more important to the range of the sensor. This is an improvement over previous research that uses checkerboard style RFID placement or massive coverage to provide information. This is important as signal interference can be detrimental when the count of the nodes is overwhelming.

The multi-story nature of the environment has been overcome as an obstacle because the simulation creates a virtual grid of each locality (e.g. building floor). The nominal pricing of the cost of Wifi and RFID sensor nodes, and the improvement found by adjusting the signal suggests it is economically feasible to test this in a real-world environment.

Descriptive statistics (RFID 10):

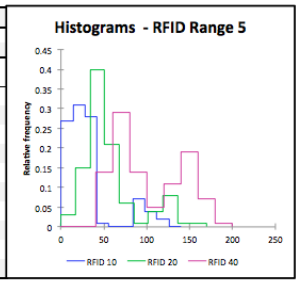
Min:	Max:	Mean:	St Dev:	
0	121	31	29	
Lower bound	Upper bound	Freq	Relative freq	Density
0	14	27	0.27	0.02
14	28	31	0.31	0.02
28	42	28	0.28	0.02
42	56	1	0.01	0.00
56	70	0	0.00	0.00
70	84	0	0.00	0.00
84	98	7	0.07	0.01
98	112	4	0.04	0.00
112	126	2	0.02	0.00
126	140	0	0.00	0.00

Descriptive statistics (RFID 20):

Min:	Max:	Mean:	St Dev:	
14	155	56	32	
Lower bound	Upper bound	Freq	Relative freq	Density
0	17	3	0.03	0.00
17	34	15	0.15	0.01
34	51	40	0.40	0.02
51	68	21	0.21	0.01
68	85	6	0.06	0.00
85	102	1	0.01	0.00
102	119	4	0.04	0.00
119	136	8	0.08	0.00
136	153	1	0.01	0.00
153	170	1	0.01	0.00

Descriptive statistics (RFID 40):

Min:	Max:	Mean:	St Dev:	
48	181	102	39	
Lower bound	Upper bound	Freq	Relative freq	Density
0	20	0	0.00	0.00
20	40	0	0.00	0.00
40	60	14	0.14	0.01
60	80	29	0.29	0.01
80	100	14	0.14	0.01
100	120	5	0.05	0.00
120	140	11	0.11	0.01
140	160	19	0.19	0.01
160	180	7	0.07	0.00
180	200	1	0.01	0.00



Descriptive statistics (RFID 10):

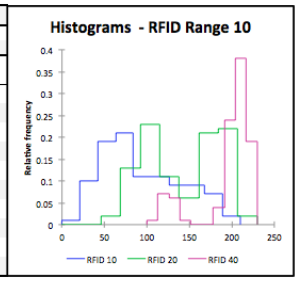
Min:	Max:	Mean:	St Dev:	
12	195	95	46	
Lower bound	Upper bound	Freq	Relative freq	Density
0	21	1	0.01	0.00
21	42	10	0.10	0.00
42	63	19	0.19	0.01
63	84	21	0.21	0.01
84	105	11	0.11	0.01
105	126	11	0.11	0.01
126	147	9	0.09	0.00
147	168	9	0.09	0.00
168	189	7	0.07	0.00
189	210	2	0.02	0.00

Descriptive statistics (RFID 20):

Min:	Max:	Mean:	St Dev:	
62	213	142	45	
Lower bound	Upper bound	Freq	Relative freq	Density
0	23	0	0.00	0.00
23	46	0	0.00	0.00
46	69	2	0.02	0.00
69	92	13	0.13	0.01
92	115	23	0.23	0.01
115	138	11	0.11	0.00
138	161	6	0.06	0.00
161	184	21	0.21	0.01
184	207	22	0.22	0.01
207	230	2	0.02	0.00

Descriptive statistics (RFID 40):

Min:	Max:	Mean:	St Dev:	
112	219	196	31	
Lower bound	Upper bound	Freq	Relative freq	Density
100	113	1	0.01	0.00
113	126	7	0.07	0.01
126	139	6	0.06	0.00
139	152	1	0.01	0.00
152	165	0	0.00	0.00
165	178	0	0.00	0.00
178	191	4	0.04	0.00
191	204	24	0.24	0.02
204	217	38	0.38	0.03
217	230	19	0.19	0.01



Descriptive statistics (RFID 10):

Min:	Max:	Mean:	St Dev:	
0	121	28	25	
Lower bound	Upper bound	Freq	Relative freq	Density
0	14	22	0.22	0.02
14	28	44	0.44	0.03
28	42	20	0.20	0.01
42	56	5	0.05	0.00
56	70	1	0.01	0.00
70	84	0	0.00	0.00
84	98	5	0.05	0.00
98	112	1	0.01	0.00
112	126	2	0.02	0.00
126	140	0	0.00	0.00

Descriptive statistics (RFID 20):

Min:	Max:	Mean:	St Dev:	
4	161	53	33	
Lower bound	Upper bound	Freq	Relative freq	Density
0	18	4	0.04	0.00
18	36	26	0.26	0.01
36	54	38	0.38	0.02
54	72	17	0.17	0.01
72	90	1	0.01	0.00
90	108	0	0.00	0.00
108	126	9	0.09	0.01
126	144	4	0.04	0.00
144	162	1	0.01	0.00
162	180	0	0.00	0.00

Descriptive statistics (RFID 40):

Min:	Max:	Mean:	St Dev:	
38	181	91	38	
Lower bound	Upper bound	Freq	Relative freq	Density
0	20	0	0.00	0.00
20	40	1	0.01	0.00
40	60	17	0.17	0.01
60	80	34	0.34	0.02
80	100	20	0.20	0.01
100	120	2	0.02	0.00
120	140	8	0.08	0.00
140	160	11	0.11	0.01
160	180	6	0.06	0.00
180	200	1	0.01	0.00

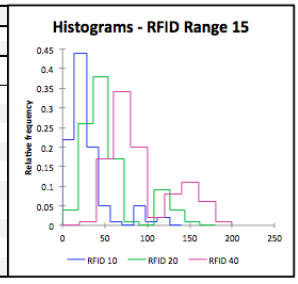


Figure 5.28.: Histogram of all RFID configurations

Descriptive statistics for GPS p-loss 60%

Min:	Max:	Mean:	St Dev:	
21	129	87	37	
Lower bound	Upper bound	Freq	Relative freq	Density
0	14	0	0.00	0.00
14	28	1	0.00	0.00
28	42	231	0.23	0.02
42	56	155	0.16	0.01
56	70	0	0.00	0.00
70	84	0	0.00	0.00
84	98	0	0.00	0.00
98	112	101	0.10	0.01
112	126	498	0.50	0.04
126	140	14	0.01	0.00

Descriptive statistics for GPS p-loss 80%

Min:	Max:	Mean:	St Dev:	
21	121	68	38	
Lower bound	Upper bound	Freq	Relative freq	Density
0	14	0	0.00	0.00
14	28	82	0.08	0.01
28	42	440	0.44	0.03
42	56	18	0.02	0.00
56	70	0	0.00	0.00
70	84	0	0.00	0.00
84	98	3	0.00	0.00
98	112	335	0.34	0.02
112	126	122	0.12	0.01
126	140	0	0.00	0.00

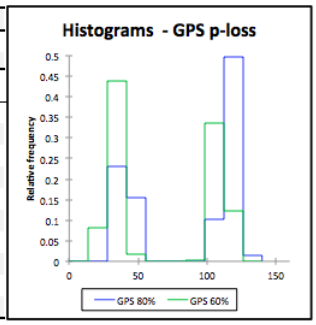


Figure 5.29.: Histogram for comparing the probabilities of GPS loss

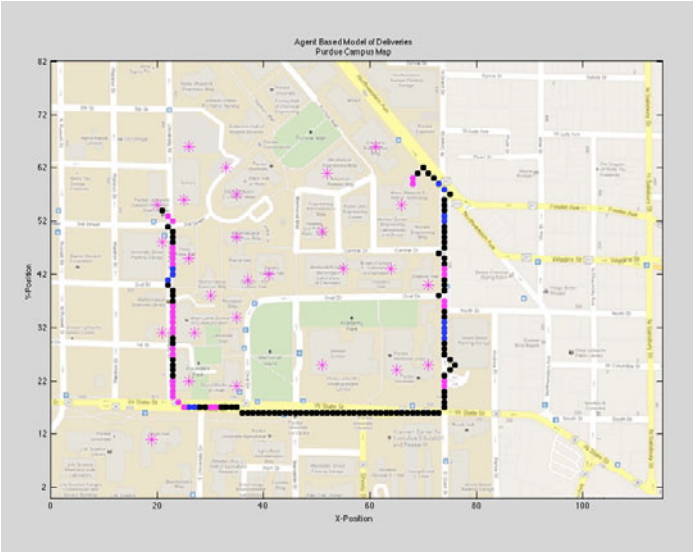


Figure 5.30.: The sensor system availability on the Purdue campus

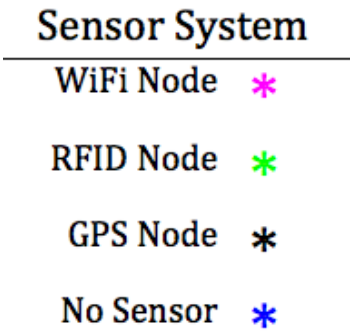


Figure 5.31.: Sensor Legend

Figure 5.32.: The sensor system availability in Knoy

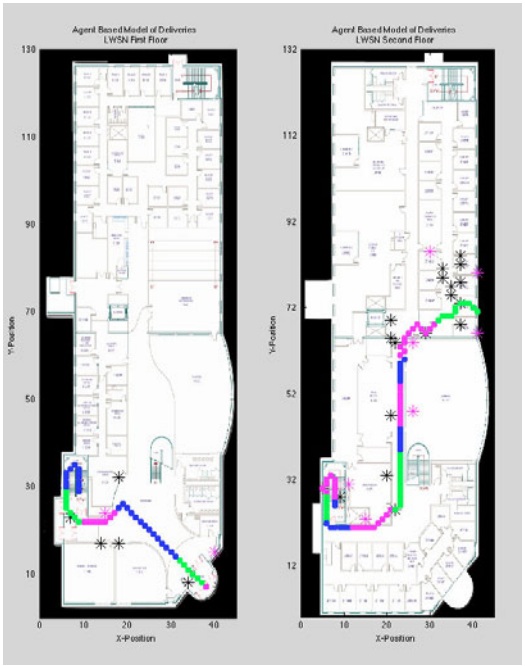


Figure 5.33.: The sensor system availability in Lawson

CHAPTER 6. CONCLUSION

6.1 Discussion

The results of the model and data were presented in the previous chapter. The three independent variables (number of sensors, range of sensors, signal combinations) produced identifiable and statistically significant improvements over previous research. The systems were combined into a classic system-of-systems framework using multi-agent organization that allowed the utilization of current technologies, while providing a common interface. This interface allowed accessibility irrespective of transportation type, location, connectivity or immediate environment. The original goals of the research were:

- Examine use of agent-based model

  This was interspersed throughout the entire research.

- Formalize theoretical description of SoS

  This was accomplished in the Methodology Chapter.

- Perform ABM

  This occurred in the Prototype, Methodology and Agent-based SoS Results Chapters.

- Implement prototypes

  Each prototype is detailed and discussed in the Prototype and Agent-based SoS Chapters.

The literature review of each topic included both the state of the art (in appropriate areas), as well as work with relevance to this research. Due to the interdisciplinary and lengthy research available to the area, the review was broken into

several subtopics, which included mobile devices, indoor localization, modeling and simulation, complex systems and path-planning. The purpose of the review was to identify current trends as well as areas open for improvement.

The data illustrated the best percentage increase in signal availability (i.e. higher time-step count) was a mid Wifi range of 10 cells and a sensor count of 20 nodes and an RFID range of 5 cells and a sensor count of 40 nodes. The nodes are not required to be distributed in a grid fashion, placement is secondary to the count and more important to the range of the sensor. This is an improvement over previous research that uses checkerboard style RFID placement or massive coverage to provide information. This is important as signal interference can be detrimental when the count of the nodes is overwhelming. The multi-story nature of the environment was handled as the simulation creates a virtual grid of each locality (e.g. building floor), while it is economically feasible to test this in a real-world environment.

6.2 Conclusion

Location-based services and navigation is a relevant topic that is pertinent to most digital device users. With this as motivation, this research studied the idea of a multi-agent system-of-systems (MaSoS) for navigation in a heterogeneous environment. An example would be moving through different environments that autonomously select available sensor systems to aid in localization. Experiments were designed to critically examine whether an agent or multi-agent system is appropriate for a navigation application. The system was formally defined and implemented through both application and model.

Typically localization and navigation are available based on the device at hand, a user's location, network coverage or mode of transportation. This study converged these ideas into a model for an autonomous system. The models and prototypes constructed combined disparate localization technologies into the possibility of a single framework solution. This research quantified measures of the system by looking at

the availability of sensor systems based on location, and the cost of movement using that system. The research expanded and refined the methodology as the prototypes were developed and tested. This approach attempted to achieve a previously unexploited deployment of existing sensor technologies within a methodological framework to more readily enable testing of new or additional technologies.

The research was separated into three prototypes (P_I - P_{III}). The first prototype studied the application of several ideas into a workable prototype, the second prototype included an applied effort and a modeling effort, and the third was strictly a model to improve the data collection process.

P_I, the Android Prototype, was implemented on an existing and publicly available device. This limited the need for additional equipment. The use of an existing device and ISO standardized equipment allowed the system to be open-source and configurable. The system was able to track a user across campus and within the building as well as provide the current floor-plan based on the user's location.

P_{II}, the iPhone, combined development of an applied system and the concept of a simulation. The application explored the embedded sensors of an unaugmented smart-phone. Using these sensors results in a known squaring error (due to the double integration). An attempt was made to categorize human movement to reset the error during each step. Unfortunately, human locomotion was too complex to be reliable and this effort was set aside. The experiment was still useful as familiarity with the environment and sensors was gained.

Additionally, the first multi-agent-based model was successfully run from the environment. The simulation was implemented as a proof-of-concept. The system was designed to be multi-agent and easily adaptable to a variety of environments and scenarios. The basic system worked well and simulated an indoor environment.

P_{III}, was a simulation performed in MatLab. Within the environment, a model was developed to study the behavior of agent and multi-agent systems. The Purdue campus environment was used to represent an urban canyon. As the system was run as a simulation, agent behavior and emergent activities were explored. The simulation

provided isolation of noise with respect to the environment and allowed control to be maintained over each variable. The model was constrained by not recording any assumptions on the system performance of an actual device. The simulation provided a framework for adding additional sensors to the system. The results proved that navigation is improved by the addition of sensor nodes, up to an upper bound, but that the greater improvement came from adjusting the range of the sensor signal.

6.2.1 Contribution of Research

The contribution of the research is in the study of the mechanics of an agent-based system of navigation in a complex environment. The details of switching between individual navigation systems and transport modes was examined specifically. Additionally, the inclusion of multiple environments (e.g. outdoor, indoor and multi-floor) was handled through a virtual grid creation method that did not include massive sensor coverage.

This research was novel as it examined fusion of existing sensor technologies with a generic framework to allow the testing of new or additional technologies. Additionally, the system looked at publicly available mobile devices, existing network and sensor networks, and the possibility of new technologies. The development of simulations and prototypes assisted in measuring the validity of the study.

6.3 Future Work

The agents, the environment, and the supporting systems were all programmed in a modular format to allow for future research avenues. A number of extensions are immediately available, while others would take careful consideration. Some extensions are exceedingly simple such as allowing the agent to select the mode of transportation, randomly selecting start and end coordinates, or having multiple agents moving simultaneously.

Table 6.1: *Comparison of Prototypes*

	P_I	P_{II}	P_{III} M_I - M_{III}	P_{III} M_{IV}
Platform	Android Nexus S	Apple iPhone	MatLab Sim	MatLab Sim
Domain	Path-plan, navigate	Path-plan	Path-plan, navigate, localize	Path-plan, navigate, localize
Method	Experiment	Experiment, MAS	MaS predicated on DEVS	MaSoS
Results	Integration of GPS, Wifi and RFID	Successful ABM, Embedded sensors too noisy	Successful ABM with GPS and Wifi	Successful MaSoS with GPS, Wifi and RFID

Others concentrate more on the structure of the environment, such as the possibility of combining the low-level metric map with a higher abstract or topographical map to interconnect them.

Some of the challenges of the simulation could be addressed such as having the sensor system recalculate the path, strictly on the information available by the current sensor system. Part of solving this problem is creating an adjacency matrix that does not hop over unknown locations.

Some of the additions would solve real-world issues with the simulation. Creating an accuracy measure of the sensor systems as an additional measure of the cost of a route would provide a more realistic weight of the path cost.

Some changes that could provide interesting avenues are in the area of the agent behavior. Introducing both cooperative and competing agents would allow for the study of the system dynamics. Allowing the agents to share information and make decisions would be a closer representation of human behavior and provide the opportunity for the agent to be more intelligent and autonomous. Sharing information offers the opportunity to incorporate additional information about environments that

other agents are experiencing, or to setup a type of social component that allows for the agents to see or locate other agents.

Agents making decisions would allow more complex behavior such as agents leaving the planned path. Additionally, real-world applications immediately exist for such a device. In the case of the Purdue campus, a hand-held device could be provided to visitors who are able to move about the campus without human interaction. Pertinent information could be provided (if desired) offering historical information about the campus.

Moving the study into mobile devices, networking, and security is also of interest. Resources are always a valuable and constrained item. Studying the effect of the implementation on live devices and networks will allow for further measures to be recorded. In the current environment, security is always relevant. There is a broad area to review in terms of the user, agent, sensor, network and the security of each.

The potential value of the extension of this research includes:

1. allowing directional acuity without a priori knowledge;
2. enhancing navigation for persons with disabilities; and
3. integrating environmental knowledge (i.e. traffic congestion avoidance, weather information integration or emergency preparedness) at the point of entry to the user.

LIST OF REFERENCES

LIST OF REFERENCES

[1] M. Maier, "Architecting principles for system-of-systems," *Systems Engineering*, vol. 1, pp. 267–284, February 1999.

[2] D. A. DeLaurentis, W. A. Crossley, and M. Mane, "A taxonomy to guide system of systems decision-making in air transportation problems," *AIAA Journal of Aircraft*, vol. in press, 2011.

[3] NPD, "http://www.connected-intelligence.com/," Feb. 2012.

[4] S. P. Hall and E. Anderson, "Operating systems for mobile computing," *J. Comput. Small Coll.*, vol. 25, no. 2, pp. 64–71, 2009.

[5] F. Lin and W. Ye, "Operating system battle in the ecosystem of smartphone industry," in *IEEC '09: Proceedings of the 2009 International Symposium on Information Engineering and Electronic Commerce*, (Washington, DC, USA), pp. 617–621, IEEE Computer Society, 2009.

[6] E. Oliver, "A survey of platforms for mobile networks research," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 12, no. 4, pp. 56–63, 2008.

[7] S. Hoseinitabatabaei, A. Gluhak, and R. Tafazolli, "udirect: A novel approach for pervasive observation of user direction with mobile phones," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pp. 74 –83, march 2011.

[8] M. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pp. 306 –311, oct. 2010.

[9] C. Barthold, K. Pathapati Subbu, and R. Dantu, "Evaluation of gyroscope-embedded mobile phones," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pp. 1632 –1638, oct. 2011.

[10] M. Anvaari and S. Jansen, "Evaluating architectural openness in mobile software platforms," in *ECSA '10: Proceedings of the Fourth European Conference on Software Architecture*, (New York, NY, USA), pp. 85–92, ACM, 2010.

[11] G. Kim and B. Hong, "Rfid-based onion skin location estimation technique in indoor environment," in *Grid and Distributed Computing* (D. Slezak, T.-h. Kim, S. S. Yau, O. Gervasi, and B.-H. Kang, eds.), vol. 63 of *Communications in Computer and Information Science*, pp. 164–175, Springer Berlin Heidelberg, 2009.

[12] F. Manzoor, Y. Huang, and K. Menzel, "Passive rfid-based indoor positioning system, an algorithmic approach," in *2010 IEEE International Conference on RFID-Technology and Applications (RFID-TA)*, pp. 112 –117, jun. 2010.

[13] Y. Zhao, Y. Liu, and L. M. Ni, "Vire: Active rfid-based localization using virtual reference elimination."

[14] J. Zhao and D. DeLaurentis, "Issues and opportunities for agent-based modeling in air transportation," in *First Conference on Infrastructure Systems and Services: Building Networks for a Brighter Future (INFRA)*, pp. 1–6, November 2008.

[15] J. Zhao and Q. Li, "A method for modeling drivers' behavior rules in agent-based traffic simulation," in *Geoinformatics, 2010 18th International Conference on*, pp. 1 –4, june 2010.

[16] A. Pradhan, E. Ergen, and B. Akinci, "Technological assessment of radio frequency identification technology for indoor localization," *Journal of Computing in Civil Engineering*, vol. 23, no. 4, pp. 230–238, 2009.

[17] F.-j. Zhu, Z.-h. Wei, B.-j. Hu, J.-g. Chen, and Z.-m. Guo, "Analysis of indoor positioning approaches based on active rfid," in *WiCOM'09: Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing*, (Piscataway, NJ, USA), pp. 5182–5185, IEEE Press, 2009.

[18] L. Ni, Y. Liu, Y. C. Lau, and A. Patil, "Landmarc: indoor location sensing using active rfid," in *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pp. 407 – 415, march 2003.

[19] J. Yuan, X. Wang, L. Dong, N. Li, F. Wang, Y. Huang, F. Sun, and Y. Wang, "Isilon-an intelligent system for indoor localization and navigation based on rfid and ultrasonic techniques," in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pp. 6625 –6630, july 2010.

[20] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with rfid technology," in *In Proceedings of IEEE International Conference on Robots and Automation (ICRA'04)*, 2004.

[21] K. Yelamarthi, D. Haas, D. Nielsen, and S. Mothersell, "Rfid and gps integrated navigation system for the visually impaired," in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1149 –1152, aug. 2010.

[22] R. Mau, N. A. Melchior, M. Makatchev, and A. Steinfeld, "Blindaid: An electronic travel aid for the blind," 2008.

[23] S. Willis and S. Helal, "Rfid information grid for blind navigation and wayfinding," in *ISWC '05: Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, (Washington, DC, USA), pp. 34–37, IEEE Computer Society, 2005.

[24] N. Ahmed, S. Butler, and U. Ramachandran, "Guardianangel: An rfid-based indoor guidance and monitoring system," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 546 –551, mar. 2010.

[25] S. R. Gandhi, "A real time indoor navigation and monitoring system for firefighters and visually impaired," Master's thesis, University of Massachusetts, Amherst, May 2011.

[26] I. Constandache, R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *In Proceedings of 2010 IEEE InfoCom*, pp. 1 –9, mar. 2010.

[27] C.-H. Hsu and C.-H. Yu, "An accelerometer based approach for indoor localization," in *Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing UIC-ATC '09*, pp. 223 –227, jul. 2009.

[28] A. Parnandi, K. Le, P. Vaghela, A. Kolli, K. Dantu, S. Poduri, and G. S. Sukhatme, "Coarse in-building localization with smartphones," in *Mobile Computing, Applications, and Services*, vol. 35 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 343–354, Springer Berlin Heidelberg, 2010.

[29] A. Ofstad, E. Nicholas, R. Szcodronski, and R. R. Choudhury, "Aampl: accelerometer augmented mobile phone localization," in *MELT '08: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, (New York, NY, USA), pp. 13–18, ACM, 2008.

[30] A. Amanatiadis, D. Chrysostomou, D. Koulouriotis, and A. Gasteratos, "A fuzzy multi-sensor architecture for indoor navigation," in *Imaging Systems and Techniques (IST), 2010 IEEE International Conference on*, pp. 452 –457, July 2010.

[31] M. Kourogi, T. Ishikawa, Y. Kameda, J. Ishikawa, K. Aoki, and T. Kurata, "Pedestrian dead reckoning and its applications," in *Let's Go Out: Research in Outdoor Mixed and Augmented Reality*, Oct. 2009.

[32] A. Dekel and E. Schiller, "Drec: exploring indoor navigation with an unaugmented smart phone," in *MobileHCI '10: Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, (New York, NY, USA), pp. 393–394, ACM, 2010.

[33] M. Hynes, H. Wang, and L. Kilmartin, "Off-the-shelf mobile handset environments for deploying accelerometer based gait and activity analysis algorithms," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 5187 –5190, sep. 2009.

[34] J. Liu, R. Chen, L. Pei, W. Chen, T. Tenhunen, H. Kuusniemi, T. Kroger, and Y. Chen, "Accelerometer assisted robust wireless signal positioning based on a hidden markov model," in *In Proceedings of 2010 IEEE/ION Position Location and Navigation Symposium (PLANS)*, pp. 488 –497, may. 2010.

[35] N. A. of Sciences, "John von neumann."

[36] A. B. Ulam, "Adam b. ulam."

[37] J. Conway and R. Guy, *The book of numbers*. Copernicus, 1996.

[38] Nobelprize.org, "The sveriges riksbank prize in economic sciences in memory of alfred nobel 2005."

[39] J. M. Epstin and R. L. Axtell, "Growing artificial societies: Social science from the bottom up," in *The MIT Press*, 1996.

[40] H. V. D. Parunak, R. Savit, and R. L. Riolo, "Agent-based modeling vs. equation-based modeling: A case study and users' guide," in *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, (London, UK), pp. 10–25, Springer-Verlag, 1998.

[41] C. M. Macal and M. J. North, "Turotial on agent-based modeling and simulation," in *Winter Simulation Conference*, pp. 2–15, 2005.

[42] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation part 2: how to model with agents," in *Proceedings of the 38th conference on Winter simulation*, WSC '06, pp. 73–83, Winter Simulation Conference, 2006.

[43] A. L. Bauer, C. A. Beauchemin, and A. S. Perelson, "Agent-based modeling of host-pathogen systems: The successes and challenges," *Information Sciences*, vol. 179, no. 10, pp. 1379 – 1389, 2009. Including Special Issue on Artificial Imune Systems.

[44] D. Sallach, N. Collier, T. Howe, and M. North, "Repast (modeling toolkit)," 2006.

[45] A. Getchell, "Agent-based modeling." Working Paper, 2008.

[46] C. Nikolai and G. Madey, "Tools of the trade: A survey of various agent based modeling platforms," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 2, p. 2, 2009.

[47] B. L. Heath, *The History, Philosophy, and Practice of Agent-Based Modeling and the Development of the Conceptual Model for Simulation Diagram*. PhD thesis, Wright State University, http://etd.ohiolink.edu/send-pdf.cgi/Heath2010.

[48] B. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*. Academic Press, 1984.

[49] M. K. Traore, "Combining devs and logic," in *In the Open International Conference on Modeling and Simulation, OICMS 2005*, 2005.

[50] B. S. S. Onggo, "Running agent-based models on a discrete-event simulator," in *The European Multidisciplinary Society for Modelling and Simulation Technology*.

[51] R. Duboz, D. Versmisse, G. Quesnel, A. Muzy, and E. Ramat, *Specification of Dynamic Structure Discrete Event Multiagent Systems*, vol. 38, p. 103. Society for Computer Simulation; 1999, 2006.

[52] D. Gianni, "Bringing discrete event simulation concepts into multi-agent systems," in *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, pp. 186 –191, april 2008.

[53] D. D. Bochtis, C. G. Sørensen, and S. G. Vougioukas, "Original paper: Path planning for in-field navigation-aiding of service units," *Comput. Electron. Agric.*, vol. 74, pp. 80–90, October 2010.

[54] J. M. Usher and L. Strawderman, "Simulating operational behaviors of pedestrian navigation," *Comput. Ind. Eng.*, vol. 59, pp. 736–747, November 2010.

[55] X. Wei, M. Xiong, X. Zhang, and D. Chen, "A hybrid simulation of large crowd evacuation," in *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pp. 971 –975, dec. 2011.

[56] J. Falco, M. Idiago, A. Delgado, A. Marco, A. Asensio, and D. Cirujano, "Indoor navigation multi-agent system for the elderly and people with disabilities," in *Trends in Practical Applications of Agents and Multiagent Systems* (Y. Demazeau, F. Dignum, J. Corchado, J. Bajo, R. Corchuelo, E. Corchado, F. Fernández-Riverola, V. Julián, P. Pawlewski, and A. Campbell, eds.), vol. 71 of *Advances in Soft Computing*, pp. 437–442, Springer Berlin / Heidelberg, 2010.

[57] A. P. Y. Morere, "A multi-agent control structure for intelligent wheelchair and aide navigation for disabled people," in *In Proc. of the 2004 International Symposium on Robotics (ISR 2004)*, p. 64, March 2004.

[58] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley and Sons Ltd., 2002.

[59] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[60] G. Weiss, *Multi-agent Systems: A modern approach to distributed artificial intelligence.* Cambridge, MA: MIT Press, 1999.

[61] C. Tao and S. Huang, "An extensible multi-agent based traffic simulation system," in *Measuring Technology and Mechatronics Automation, 2009. ICMTMA '09. International Conference on*, vol. 3, pp. 713 –716, April 2009.

[62] T. Abdelaziz, M. Elammari, and C. Branki, "Masd: Towards a comprehensive multi-agent system development methodology," in *On the Move to Meaningful Internet Systems: OTM 2008 Workshops* (R. Meersman, Z. Tari, and P. Herrero, eds.), vol. 5333 of *Lecture Notes in Computer Science*, pp. 108–117, Springer Berlin / Heidelberg, 2008.

[63] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard, "Adelfe: A methodology for adaptive multi-agent systems engineering," in *Engineering Societies in the Agents World III* (P. Petta, R. Tolksdorf, and F. Zambonelli, eds.), vol. 2577 of *Lecture Notes in Computer Science*, pp. 70–81, Springer Berlin / Heidelberg, 2003.

[64] B. An, V. Lesser, and K. Sim, "Strategic agents for multi-resource negotiation," *Autonomous Agents and Multi-Agent Systems*, pp. 1–40, 2010.

[65] E. Matson, "Embedding intelligent agents to enable physical robotic and sensor organizations," in *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*, pp. 309 –315, December 2009.

[66] E. T. Matson, "Transition process distinction in multiagent organization," in *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on*, vol. 2, pp. 527 – 532, September 2009.

[67] E. Argente, V. Julian, and V. Botti, "Mas modeling based on organizations," in *Agent-Oriented Software Engineering IX* (M. Luck and J. Gomez-Sanz, eds.), vol. 5386 of *Lecture Notes in Computer Science*, pp. 16–30, Springer Berlin / Heidelberg, 2009.

[68] M. Wood and S. DeLoach, "An overview of the multiagent systems engineering methodology," in *Agent-Oriented Software Engineering* (P. Ciancarini and M. Wooldridge, eds.), vol. 1957 of *Lecture Notes in Computer Science*, pp. 1–53, Springer Berlin / Heidelberg, 2001.

[69] W. Oyenan and S. DeLoach, "Design and evaluation of a multiagent autonomic information system," in *Intelligent Agent Technology, 2007. IAT '07. IEEE/WIC/ACM International Conference on*, pp. 182 –188, Nov 2007.

[70] M. Cardoso Silva, A. Bicharra Garcia, and A. Conci, "A multi-agent system for dynamic path planning," in *Social Simulation (BWSS), 2010 Second Brazilian Workshop on*, pp. 47 –51, oct. 2010.

[71] K.-H. C. Wang and A. Botea, "Tractable multi-agent path planning on grid maps," in *Proceedings of the 21st international jont conference on Artifical intelligence*, (San Francisco, CA, USA), pp. 1870–1875, Morgan Kaufmann Publishers Inc., 2009.

[72] D. DeLaurentis and R. K. C. Callaway, "A system-of-systems perspective for public policy decisions," *Review of Policy Research*, vol. 21, no. 6, pp. 829–837, 2004.

[73] D. DeLaurentis, "Understanding transportation as a system-of-systems design problem," in *43rd AIAA Aerospace Sciences Meeting and Exhibit AAIA*, vol. 123, pp. 10–13, January 2005.

[74] R. V. Schaff, D. DeLaurentis, and M. D. Abraham, "Effective decision-making for dod humanitarian infrastructure projects using agent-based modeling," in *IEEE SMC International Conference on System-of-Systems Engineering*, San Antonio, TX, April 2007.

[75] D. DeLaurentis, "A taxonomy-based perspective for systems of systems design methods," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 1, pp. 86 – 91 Vol. 1, Oct 2005.

[76] M. Jamshidi, "System of systems engineering - new challenges for the 21st century," *IEEE Aerospace and Electronic Systems Magazine*, vol. 23, pp. 4 – 19, May 2008.

[77] V. Mahulkar, S. McKay, D. Adams, and A. Chaturvedi, "System-of-systems modeling and simulation of a ship environment with wireless and intelligent maintenance technologies," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 39, pp. 1255 –1270, November 2009.

[78] M. Mane and D. A. DeLaurentis, "System integration and risk propogation in aeronautical systems-of-systems," in *The 27th Congress of International Council of the Aeronautical Sciences (ICAS)*, Nice, France, September 2010.

[79] A. Ozkil, Z. Fan, J. Xiao, J. Kristensen, S. Dawids, K. Christensen, and H. Aanaes, "Practical indoor mobile robot navigation using hybrid maps," in *Mechatronics (ICM), 2011 IEEE International Conference on*, pp. 475 –480, april 2011.

[80] N. MacMillan, R. Allen, D. Marinakis, and S. Whitesides, "Range-based navigation system for a mobile robot," in *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pp. 16 –23, may 2011.

[81] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[82] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, February 1968.

[83] T. Yuksel and A. Sezgin, "An implementation of a path-planning algorithms for mobile robots on a grid based map." 2008.

[84] L. Rampant, "Application of dynamic graphs for the fastest route search in a transport network," Master's thesis, Technical University of Ostrava, 2011.

[85] H. Wu, A. Marshall, and W. Yu, "Path planning and following algorithms in an indoor navigation model for visually impaired," in *Second International Conference on Internet Monitoring and Protection, ICIMP 2007*, pp. 38 –38, jul. 2007.

[86] J.-H. Zhou and H.-Y. Lin, "A self-localization and path planning technique for mobile robot navigation," in *Intelligent Control and Automation (WCICA), 2011 9th World Congress on*, pp. 694 –699, june 2011.

[87] J. Sun and X. Li, "Indoor evacuation routes planning with a grid graph-based model," in *Geoinformatics, 2011 19th International Conference on*, pp. 1 –4, june 2011.

[88] B. Dasarathy, "Nearest neighbor (nn) norms: Nn pattern classification techniques," *IEEE Computer Society Press*, 1991.

[89] V. de Almeida and R. Guting, "Using dijkstra's algorithm to incrementally find the k-nearest neighbors in spatial network databases.," in *In SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pp. 58–62, ACM, 2006.

[90] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?," in *In Int. Conf. on Database Theory*, pp. 217–235, 1999.

[91] C. Sommer, *Approximate Shortest Path and Distance Queries in Networks*. PhD thesis, University of Tokyo, 2010.

[92] Wikipedia, "Comparison of agent-based modeling software," 2012.

[93] J. Wang, M. Tian, T. Zhao, and W. Yan, "A gps-free wireless mesh network localization approach," in *CMC '09: Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing*, (Washington, DC, USA), pp. 444–453, IEEE Computer Society, 2009.

[94] A. Kim, M. Kim, E. Puchaty, M. Sevcovic, and D. Delaurentis, "A system-of-systems framework for the improved capability of insurgent tracking missions involving unmanned aerial vehicles," in *System of Systems Engineering (SoSE), 2010 5th International Conference on*, pp. 1 –6, June 2010.

[95] S. Carcieri, S. Morris, and B. D. Perry, "Rfid technology to aid in navigation and organization for the blind and partially sighted thesis," 2009.

[96] C. Hekimian-Williams, B. Grant, X. Liu, Z. Zhang, and P. Kumar, "Accurate localization of rfid tags using phase difference," in *RFID, 2010 IEEE International Conference on*, pp. 89 –96, 2010.

[97] S. Saab and S. Nakad, "A standalone rfid indoor positioning system using passive tags," *Industrial Electronics, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –1, 2010.

[98] F. Camps, S. Harasse, and A. Monin, "Numerical calibration for 3-axis accelerometers and magnetometers," pp. 217 –221, jun. 2009.

[99] J. A. Hesch, F. M. Mirzaei, G. L. Mariottini, and S. I. Roumeliotis, "A 3d pose estimator for the visually impaired," in *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, (Piscataway, NJ, USA), pp. 2716–2723, IEEE Press, 2009.

[100] I. C. Lopes, B. Vaidya, and J. J. P. C. Rodrigues, "Sensorfall - an accelerometer based mobile application," pp. 1 –6, dec. 2009.

[101] P. Silva, M. Paralta, R. Caldeirinha, J. Rodrigues, and C. Serodio, "Traceme - indoor real-time location system," pp. 2721 –2725, nov. 2009.

[102] X. Sun and S. Koenig, "The fringe-saving a* search algorithm: a feasibility study," in *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, (San Francisco, CA, USA), pp. 2391–2397, Morgan Kaufmann Publishers Inc., 2007.

[103] X. Sun, W. Yeoh, and S. Koenig, "Dynamic fringe-saving a*," in *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, (Richland, SC), pp. 891–898, International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[104] X. Sun, W. Yeoh, and S. Koenig, "Moving target d* lite," in *AAMAS '10: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, (Richland, SC), pp. 67–74, International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[105] M. Lin, A. Sud, J. Van den Berg, R. Gayle, S. Curtis, H. Yeh, S. Guy, E. Andersen, S. Patil, J. Sewall, and D. Manocha, "Real-time path planning and navigation for multi-agent and crowd simulations," in *Motion in Games* (A. Egges, A. Kamphuis, and M. Overmars, eds.), vol. 5277 of *Lecture Notes in Computer Science*, pp. 23–32, Springer Berlin / Heidelberg, 2008.

[106] D. Ashlock, T. Manikas, and K. Ashenayi, "Evolving a diverse collection of robot path planning problems," in *IEEE Congress on Evolutionary Computation, CEC2006*, (Vancouver, BC), pp. 1837–1844, 2006.

[107] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *In 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2383–2388, 2002.

[108] S.Hasegawa and A.Kashihara, "Open-ended navigation path planning in self-directed learning on the web," in *IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA2004)*, (Lisbon, Portugal), pp. 487–490, 2004.

[109] S. DeLoach, "Engineering organization-based multiagent systems," in *Software Engineering for Multi-Agent Systems IV* (A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet, and A. Romanovsky, eds.), vol. 3914 of *Lecture Notes in Computer Science*, pp. 109–125, Springer Berlin / Heidelberg, 2006.

[110] S. DeLoach and J. Valenzuela, "An agent-environment interaction model," in *Agent-Oriented Software Engineering VII* (L. Padgham and F. Zambonelli, eds.), vol. 4405 of *Lecture Notes in Computer Science*, pp. 1–18, Springer Berlin / Heidelberg, 2007.

[111] S. DeLoach, W. Oyenan, and E. Matson, "A capabilities-based model for adaptive organizations," *Autonomous Agents and Multi-Agent Systems*, vol. 16, pp. 13–56, February 2008.

[112] E. Oliver, "A survey of mobile database caching strategies." CS 648 - Database Systems Implementation course project, August 2007.

[113] E. Bonabeau, "Agent-based modeling; methods and techniques for simulating human systems," in *Colloqium Paper: Adaptive Agents, Intelligence, and Emergent Human Organization; Capturing Complexity through Agent-Based Modeling*, vol. 99, pp. 7280–7287, PNAS, 2002.

[114] C. Castle and A. Crooks, "Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations," September 2006.

[115] N. Gilbert, "Agent-based models," in *Not Sure*, 2008.

[116] J. Schank, "Agent-based modeling," *agent-based-models.com*, March 2010.

[117] E. Matson and R. Bhatnagar, "Properties of capability based agent organization transition," in *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, IAT '06, (Washington, DC, USA), pp. 59–65, IEEE Computer Society, 2006.

[118] E. Matson, A. Smith, and J. Dietz, "Application of adaptive wireless sensor organizations to secure spatial domains," in *Technologies for Homeland Security, 2009. HST '09. IEEE Conference on*, pp. 67 –72, May 2009.

[119] D. DeLaurentis, "Appropriate modeling and analysis for systems of systems: Case study synopses using a taxonomy," in *System of Systems Engineering, 2008. SoSE '08. IEEE International Conference on*, pp. 1 –6, June 2008.

[120] P.-C. DeLaurentis and D. DeLaurentis, "Consideration of system of systems and service systems as complimentary approaches for healthcare problems," in *System of Systems Engineering (SoSE), 2010 5th International Conference on*, pp. 1 –6, June 2010.

[121] M. Jamshidi, "Control of system of systems," in *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, pp. 1 –16, June 2009.

[122] M. Mane and D. DeLaurentis, "Impact of programmatic system interdependencies on system-of-systems development," in *System of Systems Engineering, 2009. SoSE 2009. IEEE International Conference on*, pp. 1 –6, June 2009.

[123] M. Mane and D. DeLaurentis, "Network-level metric measuring delay propagation in networks of interdependent systems," in *System of Systems Engineering (SoSE), 2010 5th International Conference on*, pp. 1 –6, June 2010.

[124] T. Ender, R. Leurck, B. Weaver, P. Miceli, W. Blair, P. West, and D. Mavris, "Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation," *Systems Journal, IEEE*, vol. 4, pp. 156 –166, June 2010.

[125] O. Sindiy, D. DeLaurentis, K. Akaydin, and D. Smith, "Improved decision support in space exploration via system-of-systems analysis," in *System of Systems Engineering, 2007. SoSE '07. IEEE International Conference on*, pp. 1 –6, April 2007.

[126] N. Forum, "http://www.nfc-forum.org/home/," 2011.

[127] ISO, "http://www.iso.org/iso/home.html," 2011.

[128] A. C. API, "Coremotion reference," October 2010.

[129] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Dijkstra's algorithm," in *Introduction to Algorithms 2nd edition*, ch. 24, pp. 595–599, MIT Press, 1959.

[130] R. L. Knoblauch, M. T. Pietrucha, and M. Nitzburg, "Field studies of pedestrian walking speed and start-up time," *Transportation Research Board*, no. 1538, 2005.

[131] J. Bang-Jensen and G. Z. Gutin, "Distances," in *Digraphs*, Springer Monographs in Mathematics, pp. 87–126, Springer London, 2009.

[132] Cisco, "20 myths of wi-fi interference," 2012.

[133] G. Gartner and K. Rehrl, *Location based services and telecartography II: from sensor fusion to context models*. No. v. 2 in Lecture notes in geoinformation and cartography, Springer, 2008.

[134] R. L. Plackett, "Karl pearson and the chi-squared test," *International Statistical Review / Revue Internationale de Statistique*, vol. 51, no. 1, pp. pp. 59–72, 1983.

[135] B. Chalmer, *Understanding statistics*. Statistics Textbooks and Monographs, M. Dekker, 1986.

[136] M. Youssef, M. Yosef, and M. El-Derini, "Gac: Energy-efficient hybrid gps-accelerometer-compass gsm localization," in *in Arxiv preprint arXiv:1004.3174*, 2010.

[137] W. Kang, S. Son, and J. Stankovic, "Design, implementation, and evaluation of a qos-aware real-time embedded database," in *In Submission to IEEE Transactions on Computers*, 2010.

[138] S.-W. Lee and W. Kim, "On flash-based dbmss: Issues for architectural reexamination," in *In Proceedings of Journal of Object Technology*, vol. 6, pp. 39–49, 2007.

[139] BlackBerry, "http://na.blackberry.com/eng/developers/," October 2010.

[140] A. iPhone, "http://developer.apple.com/devcenter/ios/index.action," October 2010.

[141] Android, "http://developer.android.com," September 2010.

[142] Apple, "http://developer.apple.com/," October 2010.

[143] O. H. Alliance, "http://www.openhandsetalliance.com/," 2010.

[144] M. Corporation, "http://developer.windowsphone.com/," October 2010.

[145] P. W. Schroeder and D. Burton, "Microsoft backtracks on accessiility in new mobile operating system, commits to accessibility in future windows phone platform."

GLOSSARY

Signal Metrics A variety of metrics are used to obtain signal information of the tag. It can be done through RSS (received signal strength), TOA (time of arrival) or TDOA (time difference of arrival).

Received Signal Strength RSS is variable depending on the environment. It can suffer from multi-path attenuation. It is often measured using a probabilistic or deterministic scheme.

Time of Arrival TOA requires an initial clock synchronization between devices and line of sight between devices. A trilateral positional technique is used to determine the position.

Time Difference of Arrival TDOA is an improved version of TOA. Therefore it also requires clock synchronization and line of sight between devices. It also uses trilateral positioning. The benefit of TDOA is improved performance when the access points are located at a significant distance from the controlling base station.

Fingerprinting: Each of the metrics above is then processed using fingerprinting, a method that helps define the location more accurately. Fingerprinting is a form of pattern matching. It is accomplished by measuring the signal strength at different identified locations. Each RSS measurement should be unique based on the deployment of RFID tags. One of the hurdles of this technology is the requirement of pre-mapping and measuring buildings to create an accurate map. It performs best in a static environment, although this can be unrealistic in a utilized building.

Embedded Sensors: Some of the issues encountered when using embedded sensors are that they are mass-produced, are typically cheap and inaccurate. Additionally, as the mobile device is often making minor changes in orientation due to

human walking locomotion, the sensor points can accumulate exponential errors due to the additional noise.

Accelerometer: The accelerometer in a mobile phone is a sensor used to measure the acceleration of the device relative to free-fall. This is most commonly used to detect the device orientation so the phone can perform auto-rotation of the screen.

Compass: The compass in a mobile phone is a sensor used to "measure the magnetic field strength applied to the three axes of the compass" [136]. The result of this measurement is the ability to estimate the angle between the device and true north.

Distributed: In a distributed database, there is a database management system (DBMS) that controls the data across multiple locations. The distribution is transparent to the user and the control system must take responsibility for maintaining the integrity and replication of all data. There are many reasons a distributed database could be used: reliability; availability; performance increase; local autonomy; modularity; intermittent network connection; load distribution. The disadvantages of using a distributed database architecture are: the complexity involved; maintaining data integrity; and concurrency control.

Embedded: An embedded database in the context of a mobile phone is a database that is integrated with the application and delivers its functionality as part of the application. In addition, it is expected to provide indexing, concurrency control, logging and transactional guarantees [137]. For practicality it must maintain a small footprint as the device has constrained resources, i.e. less storage space and limited memory.

Shadow: A shadow database is very similar to a distributed database. To deal with the issues of a client/server database accessed via mobile device, a small version or copy of the database is cached locally on the mobile device. In this manner, when the network is unavailable or any other connectivity issues arise, the user simply uses the data available locally. Updates will resume once the connection is reestablished.

This inevitably gives rise to the issue of maintaining data synchronization between the systems.

Flash: Finally, and most relevant to this discussion, is the use of NAND flash technology as the storage medium of choice. Flash storage has the following characteristics: low power consumption; non-volatile; electronic (no moving parts with long seek and rotational delays); shock resistance; high performance; physical stability; small size (fewer gates, ergo, smaller and denser); light weight; and portability [138].

APPENDICES

Appendix: A

.1 Sensors

A number of sensors are available to aid in localization. A small subset are discussed in this chapter.

.2 Database

Mobile devices have limited resources for data storage. They also have many constraints not found on a server or workstation. When selecting a data storage strategy it is important to consider hardware features such as battery; memory; processor; and available storage space.

The issues with NAND flash technology that complicates it's use as the platform of database management systems are two fold.

- Asymmetric read (30 mili-second) and write (300 micro-second) speeds.
- Properties of the write operation: the entire block must be erased before new data can be written if it is to overwrite existing data [138].
- There exists a physical limit to the number of writes that can be done during the lifetime of the medium, approximately 100,000 writes before the block is listed as bad. For this reason, a maintenance function, wear-leveling or block-recycling, is utilized to balance the write operations across the chip.

.3 Mobile Platforms

Due to the constrained environment encountered when developing applications for smartphones, additional thought must given to available resources and their

management. As noted, accessing database systems from mobile devices gives rise to distinct categories of databases: distributed; embedded; and shadow. Connection constraints on mobile devices cause additional considerations: intermittent connectivity; packet delay or loss; bandwidth changes; network identification changes (i.e. if the device changes from 3G to Wifi it may appear to be coming from a new device.) Finally, these finite resources include limited power due to battery limitations; constrained memory; and generally less powerful computational ability.

The discussion of the big four mobile operating systems (BlackBerry [139], iPhone [140], Android [141] and Windows Mobile) in North America is presented with the focus on the functionality available in the operating system, while avoiding most discussions about specific hardware implementations.

This paper includes a look at the categories from Oliver's 2008 work [6]: interface selection; bluetooth; background processing; energy monitor; power saving control; memory management; persistent storage; and location sensing. That work is extended to include: development language; development environment; code signing; testing, emulation and tools; application deployment; and a brief look at compatible hardware.

.3.1 BlackBerry

In 1999, Research In Motion (RIM) introduced the Blackberry, a proprietary mobile phone. It originally functioned as a two-way pager, but is now one of the most widely deployed enterprise cell phones in North America. The BlackBerry achieved a dedicated following due in large part to its mini QWERTY keyboard. Version 6.0 of the operating system was released in the third quarter of 2010. The information provided is based on that version of the operating system. More information can be found at the Blackberry website [139].

Interface Selection: A Network API is available to view and select from available connections. Connections that currently have coverage can be selected from an array, or the API can determine whether a particular option currently has coverage.

Bluetooth: A Bluetooth API allows the application to access the Bluetooth serial port and initiate a scan for a server or client connection to a computer or other Bluetooth enabled device. Before the connection, there is a pairing procedure that requires the server device to receive text data from the client device. For the devices to communicate, they must both enter a shared key.

Background Processing: A service module can be set to start automatically when the device powers on that can send and receive messages in the background.

Energy Monitoring: There are a large number of APIs available to check various states of the battery: current level; charging status; temperature; etc.

Power Saving Control: Applications may control the screen status, back-light functionality, shutdown or wake the device [6].

Memory Management: A Java Virtual Machine (JVM) is used to manage memory. The JVM performs multiple tasks including: allocating memory, garbage collection, and automatic paging of data between SRAM and flash memory.

Persistent Storage: A number of methods are available to implement persistent storage: the file system; a database; and the persistent store. A developer can store data in files and folders using the FileConnection API. There is a Database API for a SQLite relational database. The PersistentStore API permits saving objects between device restarts.

Location Sensing: Location data is available through the Location Services. When the service is on, sensor data from the internal and external GPS receivers or geo-location is available. The Location Service has methods such as setLocationOn(), which permit the Location Service to be turned on programmatically. In keeping with best practices, it is important to prompt the user for permission to turn on location services.

Development Language: Java is the development language which includes the ability to execute in a protected run-time environment. The benefit to using Java as the development platform is it allows platform independence and is widely known.

Development Environment: The Java Development Kit (JDK) is available for application development. There is a plug-in for Eclipse, which is a popular development environment.

Code Signing: A Signing Authority Tool manages the code signing system using public-private key cryptography. The private key (used by the system) uses a password, whilte the public key (used by each API file) is appended individually. The typical sequence is the developer registers with the signing authority tool and then uses the tool to request a code signature.

Testing, Emulation and Tools: Emulators for all of the current smart phone systems of Blackberry are provided. These can be downloaded at BlackBerry by selecting the phone model, carrier and operating system version.

Application Deployment: Not to be outdone by other vendors, a BlackBerry App World has been launched. It is an online storefront that allows users to find, buy and download applications directly to the phone. As of October, 2010, the website has a limited time offer declaring 'registration and submission fees are waived.' Submitted applications must comply with the Vendor Guidelines and follow the terms and conditions of the BlackBerry SDK License Agreement.

Hardware: BlackBerry OS is designed to run only on OEM BlackBerry smart phones.

Accessibility: The accessibility functionality of the Blackberry is broken down into two subsections; hearing and vision.

- Hearing Blackberry lists a number of options for the hearing impaired user: push-delivery messaging; text messaging; instant messaging; visual cue, vibration and extended vibration options; closed captioning of media content; compatibility with hearing aids; and sound-isolating headsets.

- Vision Blackberry has provided a number of accessibility functionality for the visually impaired users; audible cues; vibration and extended vibration options; customizable fonts, reverse contrast, grey scale, grid layout settings, bright high resolution display, and browser zoom; key-tone and audio clicks (to audibly confirm when the track pad or trackball is pressed or used); assignable event sounds for device status updates; assignable ringtones; voice and speed dialing, including a built-in speakerphone; keyboard nib to find the numeric keypad by touch; and individual keys with a tactile feel.

.3.2 iPhone OS

The Apple iPhone Operating System is a proprietary system introduced in June, 2007. It is a Unix-based system that is derived from Mac OS X. As of March 11, 2011 version 4.3 will be released. Copious information can be found at the Apple developer website [142].

Interface Selection: The System Configuration framework allows the application to monitor the network state of the interfaces. The SCNetworkReachability API allows an application to determine the status of a system's current network configuration.

Bluetooth: Bluetooth is supported and available programmatically on the iPhone. Before using it, a common pairing procedure is required.

Background Processing: With the release of iOS version 4.0, background processing of third party applications is now officially supported.

Energy Monitor: Status information is available by registering selectors to receive notification regarding battery state changes.

Power Saving Control: The iPhone OS provides APIs to query a number of statuses of the device: battery capacity; battery state; voltage; charging status; etc.

Low Level Memory Management: The iOS does not provide memory management (i.e. garbage collection.) Instead, it utilizes an object reference counter

captured via a retain count. Each object is initialized with a retain count of one: when the retain count of any object drops to zero, the object is deallocated and memory is returned to the system. One important caveat is that the retain count is implemented programmatically. Incorrect handling can result in potentially serious memory leaks.

Persistent Storage: Apple offers Core Data as a framework for managing persistent storage. The framework offers a number of features so that the programmer does not have to implement, or optimize them separately. It is a mature code base that is highly optimized and regularly tested by Apple. The relational database engine SQLite, an embedded SQL database, is also provided by the OS.

Location Sensing: The Core Location framework uses available hardware to determine the device's current location. It does this through a combination of cellular, Wifi and GPS services.

Development Language: The programming language for the iPhone is Objective-C and lower-level c. The API has the following abstraction layers: Cocoa Touch; Media; Core Services and Core OS. The iPhone uses the model view controller (MVC) design pattern, while allowing preference and data storage in XML format.

Development Environment: The iPhone SDK is offered for download through the Apple Developer Portal. A standard development license will cost 99.00 dollars. The SDK includes Xcode, an integrated development environment (IDE), Instruments and an iPhone simulator. In an academic environment, the SDK is available at no cost through the University.

Testing, Emulation and Tools: While the SDK provides a simulator to run an application, there is also a rich set of development tools included: Interface Builder; Instruments; and Shark.

- Interface Builder allows the developer to drag and drop preconfigured components onto the interface. The standard control components include switches, text fields, buttons, custom views, and many more. The interface contents are saved to a nib file, which is a custom resource file format.

- Instruments is used to analyze the applications performance. Data is gathered from a running application and then presented in a graphical display called the timeline. Available performance monitors are: memory usage, disk activity, network activity, and graphics performance. Each of the metrics can be displayed individually or simultaneously allowing a better visual representation of the overall behavior of an application.
- Shark is a tool for sampling or tracing a single application or all running applications. Some of the available operations are: statistical sampling over time, system-level tracing, memory tracing, static analysis, level two cache profiling, processor bandwidth analysis and java code analysis.

Application Deployment: The Apple iPhone Application Store has revolutionized the way third party applications are available to the consumer. This is the standard to which all others strive. Before an application can be approved for sale, the developer should be sure that it follows the technical, design, and content specifications detailed in the App Store Review Guidelines and Human Interface Guidelines. Apple rejects applications that do not conform to these guidelines.

Hardware: The iOS is limited to run on the iPhone, iPod Touch and the iPad platforms only.

Accessibility: To enable comparison, Apples accessibility is also presented for both hearing and vision.

- Hearing Apple offers a variety of hearing impaired accessibility options; text messaging; instant messaging; visual cue and vibrating options; open or closed captions and subtitling for media content; TTY support with an appropriate adapter; hearing aid-compatible induction ear loop; and visual voicemail (to see all your messages at once). Mono audio allows both left and right channel sources to be combined to both ears and sound-isolating headsets are available.
- Vision The iPhone also comes with a wealth of features for the vision impaired; audible and vibration options; changeable fonts and font sizes; assignable ring-

tones; audible caller ID; built-in speakerphone; and audible alerts for phone functionality.

Perhaps the most complete offering is VoiceOver, a gesture-based screen reader included with the iPhone. This is an adaptation of the Mac software for the iPhone. Since it is a touch screen, a user can point anywhere on the screen to hear an audible indication of the item. Once the finger is pointing at the appropriate item, the user can double-tap, drag or flick to control the screen.

VoiceOver is a fully customizable experience with support for twenty-one languages. A user can adjust the speaking rate for personal preferences. Computer activity is indicated by distinct sounds to distinguish what is happening, i.e.; opening an application; updating a screen or presenting a dialog box. The volume of background activities (such as music) is automatically lowered when VoiceOver is active, so that the words are clear.

Due to a flat touch screen, there is no built-in tactile feedback for typing. Instead, VoiceOver will echo each character as it types, or can echo the word if preferred, in any native application. Word prediction and spelling correction are also available to assist in typing. For built-in applications, VoiceOver is available to perform many typical functions: place and receive calls, messaging, weather, Internet browsing, etc. For those that prefer a tactile feel, wireless Braille displays are available to control the phone.

.3.3 Android

Google purchased Android, Inc. in July 2005. The first phone with the Google-backed AOS was presented on January 5th, 2010. The Google-backed AOS is an open-source "software stack for mobile devices" [141]. It is managed through the Open Handset Alliance. [143] The current version at the time of writing is 2.3 (Gingerbread) for smartphones and 3.0 (Honeycomb) for tablet versions. Android utilizes Linux Kernel version 2.6 as a base. Android is made up of an operating system (implemented

in c), middleware and applications. The main five components of the operating system are the Linux Kernel, Libraries, the Runtime environment, Application framework and applications.

Interface Selection: A connectivity manager class returns the state of network connectivity. It also notifies applications of any status changes. The class is responsible for: monitoring Wifi, GPRS UMTS, etc.; sending notifications when the network connectivity changes; and attempting automatic network fail-over.

Bluetooth: The application framework provides support for a bluetooth class. The functionality allows an application to: scan for other bluetooth devices; check for paired devices; establish RFCOMM channels; service discovery data transfer; and connection management.

Background Processing: Background processing is permitted, and has little impact on the user experience. Processes can be terminated if memory runs low. at any time to reclaim memory for a foreground, visible, or service process. Typically, many processes run in the background. To maintain ordered chaos, the processes are kept in a LRU (least recently used) list. This allows the lease recently used process to be killed first. If a developer implements the methods correctly (including current state), killing a process should not, in theory, have any negative impact on the user experience.

Energy Monitor: A battery manager class allows the status of the battery to be obtained.

Power Saving Control: Power saving controls are included in the platform. Third party applications may query the battery level, AC charging state, schedule and configure energy saving features. The developer may put the device to sleep or maintain a specific power level.

Memory Management: Android uses its own run time and VM to manage application memory. Rather than a traditional Java Virtual Machine (JVM), Android uses a custom VM. The VM is designed to efficiently run multiple instances on a single

device. This scheme allows the application to relinquish all responsibility for memory and process management to the Android run time and Dalvik garbage collector.

Persistent Storage: Persistent storage is available as an embedded SQLite database or conventional file I/O.

Location Sensing: GPS and cell tower triangulation (as signal permits) are available as localization techniques.

Development Language: Applications are written in Java and bundled by the aapt tool. This file is used to distribute and install the application on mobile devices.

Development Environment: Google provides an SDK, a downloadable package that includes system files, APIs, add-ons and development or debugging tools. As Android is programmed using Java, many Java Development Tools (JDT) are included. A plug-in for Eclipse is also provided, called the Android Development Tools (ADT). ADT includes the Dalvik Debug Monitor Server (DDMS) which has many capabilities: capture screenshots; manage port-forwarding; set breakpoints; view thread and process information.

Code Signing: Code signing is a requirement for applications. A digital signature is required, with the private key held by the application developer. Trust relationships are established through certification, however self-signing of certificates is permitted.

Testing, Emulation and Tools: The Android SDK includes a large set of tools to debug and test applications including: Traceview; Mksdcard; UI/Application Exerciser Monkey; Dalvik Debug Monitor Service; QEMU-based emulator; and the Android Debug Bridge.

- Traceview is a GUI provided interface to view trace file data. This includes viewing the results of method calls and their run times to help you profile the performance of the application.
- Mksdcard allows the developer to create a disk image for the emulator to simulate an external storage device, i.e. an SD Card.

- The User Interface and Application Exerciser Monkey allows the developer to generate streams of events, clicks and touches.
- The Dalvik Debug Monitor Service (DDMS) is a virtual machine whose functionality allows management and debugging of processes through the emulator or device. The management allows a developer to select specific processes for generating traces, and viewing heap or thread information for the selected process.
- A QEMU-based emulator is included with Android SDK. The emulator lets you prototype, develop, and test Android applications without using a physical device. All functionality is available on the emulator, except for placing actual phone calls (obviously.)
- The emulator makes use of the Android Virtual Devices (AVD). This allows a developer to setup different configurations of hardware and software to test their application against.
- The Android Debug Bridge (ADB) offers device management cooperation to the emulator: moving and syncing files; port forwarding; and a UNIX shell. It also permits command line access to the device or the simulated device.

Application Deployment: Android Market enables developers to easily publish and distribute their applications directly to users of Android-compatible phones. To publish applications to the Android Market, a developer must: create a developer profile; pay a 25.00 dollar registration fee; and agree to the Android Market Developer Distribution Agreement.

Hardware: Multiple hardware vendors offer the Google AOS on their hardware, as it is managed through the Open Handset Alliance consortium (to which many hardware vendors belong.)

Accessibility: Accessibility functionality is harder to track down on Google Android, as often features change depending on the hardware platform. The functionality is subcategorized into hearing and vision.

- Hearing Android supports many standard functions such as messaging, vibration and visual cues.
- Vision There are a variety of tools available for the Android platform. A standardized Text-To-Speech API is part of the Android SDK and works in the following languages: English; French; Italian; Spanish; and German. A set of accessibility APIs exist to create screen readers. Auditory and haptic options are available using forces, vibrations and motions for tactile feedback; TalkBack; SoundBack; and KickBack. Eyes-Free, provides UI enhancements for the touch-screen for better accessibility. Like the iPhone, there is an application, the vOICe, which contains advanced accessibility features, although it is CPU and battery intensive. The phone can be controlled via voice commands. If the Text-To-Speech library is installed, the vOICe functions as a complete screen-reader. The compass provides heading changes while moving, in addition to a talking locator that speaks the street names and intersections close to the user. Additionally, walking directions are available via Google Navigation. The camera view is used with an auditory system table to provide live feedback about the current environment. It may be augmented with special fish-eye lenses for a broader view. The vOICe software is currently hampered more by the limitations of the hardware platform, and it will continue to evolve as the hardware permits.

.3.4 Windows Phone 7

Microsoft announced Windows Phone 7 in February, 2010 at Mobile World Congress in Spain. The public release date for Windows Phone 7 was October 11, 2010 with the first device appearing in November, 2010. Technical specifications shown below are found at [144].

Interface Selection: Interface selection and status is available through the GetIsNetworkAvailable method.

Bluetooth: The official word is that the Bluetooth API's continue to be unavailable to developers through the first version release.

Background Processing: Background processing is permitted by native applications, however third party applications will not have this ability. This is similar to earlier version of the Apple iOS. It is expected as the platform improves battery life, network utilization and application predictability that this support will eventually be available for third party applications.

Energy Monitor: There is currently no API available to access this information.

Power Saving Control: There is currently no API available to power up or down any portion of the phone.

Memory Management: Windows Phone 7 is built on managed code and comes with built-in garbage collection.

Persistent Storage: There is no access to the file system. The only available space is the isolated (and sandboxed) storage.

Location Sensing: GPS data is available through the Location service, a managed API that can use input from Wifi or GPS.

Development Language: C# is the only currently support development language. Microsoft admits to interest in developing in Visual Basic, C++ or other .Net apps, and support for these may be added over time.

Development Environment: The development environment provided is Visual Studio 2010 Express, an integrated development environment (IDE). It includes Silverlight and XNA frameworks. It also includes standard IDE functions: designing; debugging; packaging, etc.

Code Signing: If the submitted application passes certificating testing (barring any failures), the code signing process is automatic. Microsoft uses Authenticode certificates which must be installed and available for applications to run on mobile devices.

Testing, Emulation and Tools: The set of tools bundled together include: the Windows Phone Emulator; XNA Game Studio; and Expression Blend.

- The Emulator provides a virtualized environment to run developed applications prior to live testing.
- The XNA Game Studio extends Visual Studio's XNA framework to allow game designers to create games for Windows Phone or Xbox 360. It includes tools for easier inclusion of graphic and audio content into games.
- Expression Blend is meant to aid in creative Silverlight-based applications using XAML-based interfaces.

Application Deployment: A Windows Live ID is required and is available for sign-up with no cost. Developers must agree to and sign the application provider agreement. There is an annual subscription fee of 99.00 dollars (the same as the Apple Developer fee).

Developers can publish application via Windows Phone Marketplace. Requirements include submitting a .xap file containing all the relevant materials for certification. The certification process verifies the applications behavior.

Once the package has satisfied the Windows Phone Marketplace certification requirements, the developer is notified and can publish the application to the Windows Phone Marketplace through the developer portal. Applications are then made available for consumers to download in the Windows Phone Marketplace. Credit card and mobile-operator billing operations are supported.

Hardware: The Windows Phone 7 system does not require a proprietary system for deployment, however Microsoft does have a list of minimum system requirements that manufacturers must agree to. They wants to enforce a consistent set of hardware capabilities for Windows Phone 7, such that the hardware will be made up of common and identifiable hardware elements.

Accessibility: The accessibility features of the Windows Phone 7 are listed below.

- Hearing For those with hearing impairments, TTY and specific hearing aid support is available.
- Vision For visual assistance, Windows Phone 7 offers the following customizations: ability to adjust brightness and contrast; customizable visual schemes, speech recognition and many standard phone functions can be controlled by voice. Other than these brief offerings, due to the complete overhaul of the underlying operating system, much of the previous accessibility available in earlier versions, is no longer applicable. Reworking the functionality is expected to be a "multi-year process." [145]

.3.5 Summary of Mobile Devices

The top four mobile devices in North America that were discussed above, are summarized in Fig: 2.3.

(Note: A version of portions of this chapter was presented at the conference *ASEE Global Colloquium on Engineering Education 2011* in June, 2011. It was authored by Teresa A. Shanklin and Kyle D. Lutes)

Appendix: B

.4 Matlab Source Code

Main.m

```
%main.m
%Agent Based Model - Indoor Navigation

 clc; close all; %clear all;
p=0;
num=100;

for p=1:num

    global fig1 fig2 fig3 NoWifi NoRfid Targ Path
CMap1 Nav1 CMap2 Nav2 CMap3
Nav3 CMap4 Nav4 CMap5 Nav5 test
GPSProb WifiDist RfidDist;
    warning('off','MATLAB:xlsread:Mode');



    NoWifi = 80;
    NoRfid = 80;
    GPSProb = 60;
    WifiDist = 10;
    RfidDist = 5;
```

```
    build1NodeMapKNOY1;
    build2NodeMapKNOY2;
    build3NodeMapPUMAP;
    build1NodeMapLWSN1;
    build2NodeMapLWSN2;

% Parameters
    Targ.num = 1;    %Number of targets
    Targ.End = 0;
    SpeedTotal = 0;
    TimeTotal = 0;
    Total = 0;

% Setup Initial Conditions
    initializeTargArray();
    initializeGoalArray();
    findShortestPath(Targ.State(7:8),Targ.State(12:13),
CMap2.node,CMap2.adjmat);

%Targ.History = cell(1,Targ.num);
    History = Targ.State(1:29);

    fprintf('Agents intialized...\n');

   loadFigures();
    % Iteration
    t=1;
    k=1;
```

```
while (Targ.End ~= Targ.num)
    t = t+ 1;

    if (mod(t,2) == 0)
        updateTargState(t);
    end

%update position
    updatePos(t);

   Targ.State(29) = Targ.State(21) + Targ.State(26) + Targ.State(28);

   % Save history
    History = [History, Targ.State(1:29)];

    SpeedTotal = SpeedTotal + Targ.State(26);
    TimeTotal = TimeTotal + Targ.State(28);
    Total= Total + Targ.State(29);

   % ***********PLOT***************
    %Record system state at time 't'
    %getPlotFrame();
    %frameArray(t) = getframe;
end

dt = sprintf('Run%d', p);
dt = horzcat(dt, datestr(now, '--mm-dd-yy--HH:MM:SS'));

file = horzcat(horzcat('/Matlab/Sim/hist-',dt),'.csv');
```

```
 xlswrite(file, History,1)

Details0 = {
     Total,
     SpeedTotal,
     TimeTotal,
     NoWifi,
     NoRfid,
     GPSProb,
     WifiDist,
     RfidDist};

 if (p == 1)
     Details = Details0;
 else
     Details2 = Details;
     Details = horzcat(Details2, Details0);
 end


 figure(fig3);
 subplot(1,2,1);
 figFrame = getframe;
 file= horzcat(horzcat('/Matlab/Sim/LWSN1-',dt),'.png');
 imwrite(figFrame.cdata, file);
 subplot(1,2,2);
 figFrame = getframe;
 file= horzcat(horzcat('/Matlab/Sim/LWSN2-',dt),'.png');
 imwrite(figFrame.cdata, file);
```

```
    figure(fig2);
    figFrame = getframe;
    file = horzcat(horzcat('/Matlab/Sim/PU-',dt),'.png');
    imwrite(figFrame.cdata, file);

    figure(fig1);
    subplot(1,2,2);
    figFrame = getframe;
    file = horzcat(horzcat('/Matlab/Sim/KNOY2-',dt),'.png');
    imwrite(figFrame.cdata, file);
    subplot(1,2,1);
    figFrame = getframe;
    file = horzcat(horzcat('/Matlab/Sim/KNOY1-',dt),'.png');
    imwrite(figFrame.cdata, file);

clear global fig1 fig2 fig3 NoWifi NoRfid Targ CMap1
Nav1 CMap2 Nav2 CMap3 Nav3 CMap4
Nav4 CMap5 Nav5 test;
    clear History Total dt figFrame file frameArray t Details0
Details1 GPSProb RfidDist WifiDist k;
    close(gcf);
    close(gcf);
    close(gcf);

    p= p+1;
    fprintf('Run number \%d\n', p);
end\
```

```
Details1 = {...
    'Total',...
    'SpeedTotal',...
    'TimeTotal',...
    'NumWifi',...
    'NumRfid',...
    'GPSProb',...
    'WifiDist',...
    'RfidDist',...
    };

    Details1 = reshape(Details1,8,1);
    Details3 = horzcat(Details1, Details);
    dt = datestr(now, '--mm-dd-yy--HH:MM:SS');
    file = horzcat(horzcat('/Matlab/Sim/hist-Summary-',dt),'.csv');
    cell2csv(file, Details3, ',');

beep;

%Offer the choice of saving the movie to file
choiceSaveMovie(frameArray);
```

build1NodeMapKNOY1

```
function build1NodeMapKNOY1

% Read in xls file and convert to a node list
global NoWifi NoRfid CMap1 Nav1;
[num,txt,CMap1] = xlsread('Matlab/CMap','KNOY\_1flr','','basic');

CMap1 = CMap1(1:146, 1:51);
fprintf('Excel cost map read in...\n');

numNodes = size(CMap1,1)*size(CMap1,2);
rowL = size(CMap1,2);

CMap1.node = Inf(numNodes,3);

row = 1;
col = 1;

for i = 1:(numNodes-1)
    CMap1.node(i,:) = [col row CMap1{row,col}];
    col = col + 1;
    if col > rowL
        row = row + 1;  %Move to next row
        col = 1;
    end
end

% Find buildings
b = find(CMap1.node(:,3)=='B');
```

```
Nav1.buildings = CMap1.node(b,1:2);
CMap1.node(b,3) = Inf; %set to infinity - cannot travel

%Find doors
d = find(CMap1.node(:,3)=='D');
Nav1.doors = CMap1.node(d,1:2);

%Find move to next flr
n = find(CMap1.node(:,3)=='N');
Nav1.nextflr = CMap1.node(n,1:2);

%Find WIFI Sources
w = find(CMap1.node(:,3) == 'W');
Nav1.wifi = CMap1.node(w,1:2);

%Find RFID Sources
r = find(CMap1.node(:,3) == 'R');
Nav1.rfid = CMap1.node(r,1:2);

%Find outside of buildings
o = find(CMap1.node(:,3)=='O');
Nav1.outside = CMap1.node(o,1:2);
CMap1.node(o,3) = Inf;  %set to infinity - this is inside map

CMap1.node(CMap1.node(:,3)=='I',3) = 4;
CMap1.node(CMap1.node(:,3)=='D',3) = 4;
CMap1.node(CMap1.node(:,3)=='N',3) = 4;
CMap1.node(CMap1.node(:,3)=='W',3) = 4;
CMap1.node(CMap1.node(:,3)=='R',3) = 4;
```

```
%Log/Maintenance
fprintf('Node database generated...\n');

% Build weighted adjacency matrix
CMap1.adjmat = Inf(numNodes);
adjmatWiFi = Inf(numNodes);

fprintf('Building weighted adjacency matrix...');

nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
for k = 1:numNodes

    switch mod(k,rowL)
        case 1
            nbor = [-rowL,-rowL+1,1,rowL,rowL+1];
        case 0
            nbor = [-(rowL+1),-rowL,-1,rowL-1,rowL];
        otherwise
            nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
    end



    for j = 1:length(nbor)
        if k + nbor(j) <= numNodes && k + nbor(j)>0
            CMap1.adjmat(k,k+nbor(j)) = CMap1.node(k+nbor(j),3);
    end
end
```

```
Nav1.wifi = zeros(NoWifi,2);
for k=1:NoWifi
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap1.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap1.node(ran_num,:);
        end
    end
    Nav1.wifi(k,:) = targetObj(1:2);  %cur xPosition
end

Nav1.rfid = zeros(NoRfid,2);
for k=1:NoRfid
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap1.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap1.node(ran_num,:);
```

```
        end
    end
    Nav1.rfid(k,:) = targetObj(1:2);  %cur xPosition
end
fprintf('Done\n');
end
```

build1NodeMapLWSN1

```
function build1NodeMapLWSN1

global NoWifi NoRfid CMap4 Nav4;
% Read in xls file and convert to a node list
[num,txt,CMap1] = xlsread('/Matlab/CMap','LWSN_1flr','','basic');

CMap1 = CMap1(1:127, 1:45);
fprintf('Excel cost map read in...\n');

numNodes = size(CMap1,1)*size(CMap1,2);
rowL = size(CMap1,2);

CMap4.node = Inf(numNodes,3);

row = 1;
col = 1;

for i = 1:(numNodes-1)
    CMap4.node(i,:) = [col row CMap1{row,col}];
    col = col + 1;
    if col > rowL
        row = row + 1;  %Move to next row
        col = 1;
    end
end

% Find buildings
b = find(CMap4.node(:,3)=='B');
```

```
Nav4.buildings = CMap4.node(b,1:2);
CMap4.node(b,3) = Inf; %set to infinity - cannot travel

%Find doors
d = find(CMap4.node(:,3)=='D');
Nav4.doors = CMap4.node(d,1:2);

%Find move to next flr
n = find(CMap4.node(:,3)=='N');
Nav4.nextflr = CMap4.node(n,1:2);

%Find outside of buildings
o = find(CMap4.node(:,3)=='O');
Nav4.outside = CMap4.node(o,1:2);
CMap4.node(o,3) = Inf;  %set to infinity - this is inside map

CMap4.node(CMap4.node(:,3)=='I',3) = 4;
CMap4.node(CMap4.node(:,3)=='D',3) = 4;
CMap4.node(CMap4.node(:,3)=='N',3) = 4;
CMap4.node(CMap4.node(:,3)=='W',3) = 4;
CMap4.node(CMap4.node(:,3)=='R',3) = 4;

%Log/Maintenance
fprintf('Node database generated...\n');

% Build weighted adjacency matrix
CMap4.adjmat = Inf(numNodes);
fprintf('Building weighted adjacency matrix...LWSN 1');
```

```
nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
for k = 1:numNodes

    switch mod(k,rowL)
        case 1
            nbor = [-rowL,-rowL+1,1,rowL,rowL+1];
        case 0
            nbor = [-(rowL+1),-rowL,-1,rowL-1,rowL];
        otherwise
            nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
    end

    for j = 1:length(nbor)
        if k + nbor(j) <= numNodes && k + nbor(j)>0
            CMap4.adjmat(k,k+nbor(j)) = CMap4.node(k+nbor(j),3);
        end
    end
end



%SETUP WIFI
Nav4.wifi = zeros(NoWifi,2);
for k=1:NoWifi

    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap4.node(ran_num,3) == infCheck)
```

```
  %make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap4.node(ran_num,:);
        end
    end
    Nav4.wifi(k,:) = targetObj(1:2);  %cur xPosition
end

Nav4.rfid = zeros(NoRfid,2);
for k=1:NoRfid
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap4.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap4.node(ran_num,:);
        end
    end
    Nav4.rfid(k,:) = targetObj(1:2);  %cur xPosition
end

fprintf('Done\n');
end
```

build2NodeMapKNOY2

```
function build2NodeMapKNOY2

global NoWifi NoRfid CMap2 Nav2;
% Read in xls file and convert to a node list
[num,txt,CMap2] = xlsread('/Matlab/CMap','KNOY_2flr','','basic');

CMap2 = CMap2(1:146, 1:51);
fprintf('Excel cost map read in...\n');

%% Build node database from
    %node = [x y cost]

numNodes = size(CMap2,1)*size(CMap2,2);
rowL = size(CMap2,2);

CMap2.node = Inf(numNodes,3);

row = 1; col = 1;

for i = 1:(numNodes-1)

    CMap2.node(i,:) = [col row CMap2{row,col}];

    col = col + 1;

    if col > rowL
        row = row + 1;  %Move to next row
        col = 1;
```

```
    end
end

% Find buildings
b = find(CMap2.node(:,3)=='B');
Nav2.buildings = CMap2.node(b,1:2);
CMap2.node(b,3) = Inf;

%Find doors
d = find(CMap2.node(:,3)=='D');
Nav2.doors = CMap2.node(d,1:2);

%Find move to next flr
n = find(CMap2.node(:,3)=='N');
Nav2.nextflr = CMap2.node(n,1:2);

%Find outside of buildings
o = find(CMap2.node(:,3)=='O');
Nav2.outside = CMap2.node(o,1:2);
CMap2.node(o,3) = Inf;


CMap2.node(CMap2.node(:,3)=='I',3) = 4;
CMap2.node(CMap2.node(:,3)=='D',3) = 4;
CMap2.node(CMap2.node(:,3)=='N',3) = 4;
CMap2.node(CMap2.node(:,3)=='R',3) = 4;

%Log/Maintenance
fprintf('Node database generated...\n');
```

```
%% Build weighted adjacency matrix
CMap2.adjmat = Inf(numNodes);

fprintf('Building weighted adjacency matrix... KNOY2');

nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
for k = 1:numNodes

    switch mod(k,rowL)
        case 1
            nbor = [-rowL,-rowL+1,1,rowL,rowL+1];
        case 0
            nbor = [-(rowL+1),-rowL,-1,rowL-1,rowL];
        otherwise
            nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
    end

       for j = 1:length(nbor)
           if k + nbor(j) <= numNodes && k + nbor(j)>0
               CMap2.adjmat(k,k+nbor(j)) = CMap2.node(k+nbor(j),3);
           end
       end
end

%WIFI
Nav2.wifi = zeros(NoWifi,2);
for k=1:NoWifi
    infCheck = Inf;
```

```
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap2.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap2.node(ran_num,:);
        end
    end
    Nav2.wifi(k,:) = targetObj(1:2);  %cur xPosition
end

Nav2.rfid = zeros(NoRfid,2);
for k=1:NoRfid
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap2.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap2.node(ran_num,:);
        end
    end
    Nav2.rfid(k,:) = targetObj(1:2);  %cur xPosition
```

```
end
fprintf('Done\n');
end
```

build2NodeMapLWSN2

```
function build2NodeMapLWSN2

global NoWifi NoRfid CMap5 Nav5;
% Read in xls file and convert to a node list
[num,txt,CMap1] = xlsread('/Matlab/CMap','LWSN_2flr','','basic');

CMap1 = CMap1(1:131, 1:48);
fprintf('Excel cost map read in...\n');

numNodes = size(CMap1,1)*size(CMap1,2);
rowL = size(CMap1,2);

CMap5.node = Inf(numNodes,3);

row = 1;
col = 1;

for i = 1:(numNodes-1)
    CMap5.node(i,:) = [col row CMap1{row,col}];
    col = col + 1;
    if col > rowL
        row = row + 1;  %Move to next row
        col = 1;
    end
end

% Find buildings
b = find(CMap5.node(:,3)=='B');
```

```
Nav5.buildings = CMap5.node(b,1:2);
CMap5.node(b,3) = Inf; %set to infinity - cannot travel

%Find doors
d = find(CMap5.node(:,3)=='D');
Nav5.doors = CMap5.node(d,1:2);

%Find move to next flr
n = find(CMap5.node(:,3)=='N');
Nav5.nextflr = CMap5.node(n,1:2);

%Find outside of buildings
o = find(CMap5.node(:,3)=='O');
Nav5.outside = CMap5.node(o,1:2);
CMap5.node(o,3) = Inf;  %set to infinity - this is inside map

CMap5.node(CMap5.node(:,3)=='I',3) = 4;
CMap5.node(CMap5.node(:,3)=='D',3) = 4;
CMap5.node(CMap5.node(:,3)=='N',3) = 4;
CMap5.node(CMap5.node(:,3)=='W',3) = 4;
CMap5.node(CMap5.node(:,3)=='R',3) = 4;

%Log/Maintenance
fprintf('Node database generated...\n');

%% Build weighted adjacency matrix
CMap5.adjmat = Inf(numNodes);

fprintf('Building weighted adjacency matrix... LWSN2');
```

```
nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
for k = 1:numNodes

    switch mod(k,rowL)
        case 1
            nbor = [-rowL,-rowL+1,1,rowL,rowL+1];
        case 0
            nbor = [-(rowL+1),-rowL,-1,rowL-1,rowL];
        otherwise
            nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
    end

      for j = 1:length(nbor)
        if k + nbor(j) <= numNodes && k + nbor(j)>0
            CMap5.adjmat(k,k+nbor(j)) = CMap5.node(k+nbor(j),3);
        end
      end
end
%WIFI
Nav5.wifi = zeros(NoWifi,2);
for k=1:NoWifi
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap5.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
```

```
        else
            found = 1;
            targetObj = CMap5.node(ran_num,:);
        end
    end
    Nav5.wifi(k,:) = targetObj(1:2);  %cur xPosition
end

Nav5.rfid = zeros(NoRfid,2);
for k=1:NoRfid
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap5.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap5.node(ran_num,:);
        end
    end
    Nav5.rfid(k,:) = targetObj(1:2);  %cur xPosition
end
fprintf('Done\n');
end
```

build3NodeMapPUMAP

```
function build3NodeMapPUMAP

global NoWifi NoRfid CMap3 Nav3;
% Read in xls file and convert to a node list
[num,txt,CMap3] = xlsread('/Matlab/CMap','PUmap1','','basic');

CMap3 = CMap3(1:84, 1:88);  %row by column

fprintf('Excel cost map read in...\n');

numNodes = size(CMap3,1)*size(CMap3,2);
rowL = size(CMap3,2);

CMap3.node = Inf(numNodes,3);

row = 1; col = 1;

for i = 1:(numNodes-1)
    CMap3.node(i,:) = [col row CMap3{row,col}];
    col = col + 1;
    if col > rowL
        row = row + 1;  %Move to next row
        col = 1;
    end
end

% Find buildings
b = find(CMap3.node(:,3)=='B');
```

```
Nav3.buildings = CMap3.node(b,1:2);
CMap3.node(b,3) = Inf;

%Find doors
d = find(CMap3.node(:,3)=='D');
Nav3.doors = CMap3.node(d,1:2);

%Find inside
n = find(CMap3.node(:,3)=='N');
Nav3.nextflr = CMap3.node(n,1:2);

%show roads
r = find(CMap3.node(:,3) == 1);
r2= find(CMap3.node(:,3) == 2);
r = sort(vertcat(r,r2));
Nav3.road = CMap3.node(r,1:2);

CMap3.node(CMap3.node(:,3)=='I',3) = 4;
CMap3.node(CMap3.node(:,3)=='D',3) = 4;
CMap3.node(CMap3.node(:,3)=='P',3) = 10;

%Log/Maintenance
fprintf('Node database generated...\n');

%% Build weighted adjacency matrix
CMap3.adjmat = Inf(numNodes);

fprintf('Building weighted adjacency matrix...PUMap');
```

```
nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
for k = 1:numNodes

    switch mod(k,rowL)
        case 1
            nbor = [-rowL,-rowL+1,1,rowL,rowL+1];
        case 0
            nbor = [-(rowL+1),-rowL,-1,rowL-1,rowL];
        otherwise
            nbor = [-(rowL+1),-rowL,-rowL+1,-1,1,rowL-1,rowL,rowL+1];
    end

    for j = 1:length(nbor)
        if k + nbor(j) <= numNodes && k + nbor(j)>0
                CMap3.adjmat(k,k+nbor(j)) =
CMap3.node(k+nbor(j),3);
        end
    end
end

%WIFI
Nav3.wifi = zeros(NoWifi,2);
for k=1:NoWifi
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap3.node(ran_num,3) == infCheck)
 %make sure the start space is valid
```

```
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap3.node(ran_num,:);
        end
    end
    Nav3.wifi(k,:) = targetObj(1:2);  %cur xPosition
end

Nav3.rfid = zeros(NoRfid,2);
for k=1:NoRfid
    infCheck = Inf;
    found = 0;
    ran_num = randi(numNodes);
    while (found == 0)
        if  (CMap3.node(ran_num,3) == infCheck)
%make sure the start space is valid
            ran_num = randi(numNodes);
        else
            found = 1;
            targetObj = CMap3.node(ran_num,:);
        end
    end
    Nav3.rfid(k,:) = targetObj(1:2);  %cur xPosition
end
fprintf('Done\n');
end
```

findShortestPath.m

```
function findShortestPath(currentPos,targetPos,node,adjmat)

global Path;
tic

currentRowPoss = find(node(:,1)==currentPos(1));
currentCol = find(node(currentRowPoss,2)==currentPos(2));
currentNode = currentRowPoss(currentCol);

targetRowPoss = find(node(:,1)==targetPos(1));
targetCol = find(node(targetRowPoss,2)==targetPos(2));
targetNode = targetRowPoss(targetCol);

if node(currentNode,3) == Inf || node(targetNode,3) == Inf
    error('Coordinate out of bounds for pathways %d %d %d or
 %d %d %d', node(currentNode,1), node(currentNode,2),
node(currentNode,3), node(targetNode,1), node(targetNode,2),
node(targetNode,3));
end

%Find shortest path
fprintf('Searching shortest path from node %d %d to node %d %d ...',
node(currentNode,1), node(currentNode,2), node(targetNode,1),
node(targetNode,2));

[sPath, cost] = dijkstra(currentNode, targetNode, adjmat);
Path = sPath;
end
```

flrCheck.m

```
function [updatedPath] = flrCheck(Targ,CMap1, CMap2,
CMap3, CMap4, CMap5, Nav1, Nav2, Nav3, Nav4, Nav5)

for k = 1: Targ.num

    currentPos = Targ.State(7:8,k);
    objectivePos = Targ.State(17:18,k);

    doorPos = Targ.Goals(1:2,k);
    outsideDoorPos = Targ.Goals(3:4,k);
    grantParking = Targ.Goals(5:6,k);
    universityParking = Targ.Goals(7:8,k);
    lwsnDoorPos = Targ.Goals(9:10,k);
    insideDoorPos = Targ.Goals(11:12,k);

    %%%%%%%%%%
    %%
    %% START INSIDE /MOVE OUTSIDE/ END INSIDE
    %%
    %%%%%%%%%%
    if ((Targ.State(1,k)== 1 && Targ.State(16,k) == 1) &&
 (Targ.State(5,k) ~= Targ.State(20,k)))
%START IN - MOVE OUT - END IN

        if (Targ.State(4,k) == 2)
%start on 2 flr KNOY - find nearest down
            idx=knnsearch(currentPos(2,1), Nav2.nextflr(:,2), 1);
            tempPos = Nav2.nextflr(idx(1,1),:);
```

```
            interPos = reshape(tempPos,2,1);
            tempPath1 = findShortestPath(currentPos,interPos,
CMap2.node,CMap2.adjmat);

            newTempPos = Nav1.nextflr(idx(1,1),:);
%KNOY 1st flr - move to door
            newCurPos = reshape(newTempPos,2,1);
            tempPath2 = findShortestPath(newCurPos, doorPos,
CMap1.node, CMap1.adjmat);

            %move outside from knoy to lawson
            tempPath3 = findShortestPath(outsideDoorPos,
lwsnDoorPos, CMap3.node, CMap3.adjmat);

            idx=knnsearch(insideDoorPos(2,1), Nav4.nextflr(:,2),1);
%move from LWSN door to nearest up
            tempPos1 = Nav4.nextflr(idx(1,1),:);
            interPos1 = reshape(tempPos1,2,1);
            tempPath4 = findShortestPath(insideDoorPos, interPos1,
CMap4.node, CMap4.adjmat);


            newTempPos2 = Nav5.nextflr(idx(1,1),:);
%move from LWSN 2nd flr to end point
            newCurPos2 = reshape(newTempPos2,2,1);
            tempPath5 = findShortestPath(newCurPos2, objectivePos,
CMap5.node, CMap5.adjmat);
            Targ.Path{k} = horzcat(tempPath1, 0, tempPath2, 0,
tempPath3, 0, tempPath4, 0, tempPath5);
```

```
        end
    end
end
updatedPath = Targ.Path;
end
```

getPlotFrame.m

```
function  getPlotFrame()

global fig1 fig2 fig3 Targ;

%Plots the system at the current time step

%Preferences
pathStreaks = 0;
%Turning path streaks on will show the path taken for
%each agent, turning it off will show only current
%positions at each time step

if pathStreaks == 1;
    hold on;
else
    hold off;
end

if (Targ.State(6,1) == 1)
    if (Targ.State(10,1) == 1) %KNOY
        if (Targ.State(9,1) == 1)
            1flr = Targ.State;
            TF1 = isempty(1flr);
            TF2 = 0;

            %%%%%%%%%%%%%%%%
            %% INSIDE  KNOY
            %%%%%%%%%%%%%%%%
```

```
    figure(fig1);

    hold on;
    subplot(1,2,1);
    if (~TF1)
        if (Targ.State(24) == 1)
            plot(1flr(7,:),1flr(8,:),'m.', 'MSize',20);
        elseif (Targ.State(25) == 1)
            plot(1flr(7,:),1flr(8,:),'g.', 'MSize',20);
        else
            plot(1flr(7,:),1flr(8,:),'b.', 'MSize',20);
        end
    end

    hold on;
    title({'Agent Based Model of Deliveries'; 'KNOY 1- flr'});
    xlabel('X-Position'); ylabel('Y-Position');

else
    2flr = Targ.State;
    TF2 = isempty(2flr);
    TF1 = 1;

    %%%%%%%%%%%%%%%%
    %% 2nd FLOOR  KNOY
    %%%%%%%%%%%%%%%%
    figure(fig1);

    hold on;
```

```
        subplot(1,2,2);
        if (~TF2)
            if (Targ.State(24) == 1)
                plot(2flr(7,:),2flr(8,:),'m.', 'MSize',20);%wifi
            elseif (Targ.State(25) == 1)
                plot(2flr(7,:),2flr(8,:),'g.', 'MSize',20);%rfid
            else
                plot(2flr(7,:),2flr(8,:),'b.', 'MSize',20);%n/a
            end
        end
        title({'Agent Based Model of Deliveries'; 'KNOY 2nd flr'});
        xlabel('X-Position'); ylabel('Y-Position');


    end
else
    %%%LWSN
    if (Targ.State(9,1) == 1)
        1flr = Targ.State;
        TF1 = isempty(1flr);
        %TF2 = 0;


        %%%%%%%%%%%%%%%%
        %% INSIDE  1st FLOOR LWSN
        %%%%%%%%%%%%%%%%
        figure(fig3);


        hold on;
        subplot(1,2,1);
        if (~TF1)
```

```
            if (Targ.State(24) == 1)
                plot(1flr(7,:),1flr(8,:),'m.', 'MSize',20);%wifi
            elseif (Targ.State(25) == 1)
                plot(1flr(7,:),1flr(8,:),'g.', 'MSize',20);%rfid
            else
                plot(1flr(7,:),1flr(8,:),'b.', 'MSize',20);%n/a
            end
        end

        hold on;
        title({'Agent Based Model of Deliveries'; 'LWSN 1- flr'});
        xlabel('X-Position'); ylabel('Y-Position');
    else
        2flr = Targ.State;
        TF2 = isempty(2flr);
        TF1 = 1;

        %%%%%%%%%%%%%%%%
        %% 2nd FLOOR  LWSN
        %%%%%%%%%%%%%%%%
        figure(fig3);

        hold on;
        subplot(1,2,2);
        if (~TF2)
            if (Targ.State(24) == 1)
                plot(2flr(7,:),2flr(8,:),'m.', 'MSize',20);%wifi
            elseif (Targ.State(25) == 1)
                plot(2flr(7,:),2flr(8,:),'g.', 'MSize',20);%rfid
```

```
                else
                    plot(2flr(7,:),2flr(8,:),'b.', 'MSize',20);%n/a
                end
            end
            title({'Agent Based Model of Deliveries'; 'LWSN 2nd flr'});
            xlabel('X-Position'); ylabel('Y-Position');
        end
    end
else
    %%%%%%%%%%%%%%%%
    %% OUTSIDE
    %%%%%%%%%%%%%%%%
    outside = Targ.State;
    TF3 = isempty(outside);

    figure(fig2);

    hold on;
    if(~TF3)
          if (Targ.State(23) == 1)
%GPS
                    plot(outside(7,:),outside(8,:),'k.', 'MSize',20);
                elseif (Targ.State(24) == 1)
%wifi
                    plot(outside(7,:),outside(8,:),'m.', 'MSize',20);
                else
%n/a
                    plot(outside(7,:),outside(8,:),'b.', 'MSize',20);
                end
```

```
    end
    title({'Agent Based Model of Deliveries'; 'Purdue Campus Map'});
    xlabel('X-Position'); ylabel('Y-Position');

hold off;
end
```

initializeGoalArray.m

```
function initializeGoalArray()

%Function initializes the states of the various intermediate goals
global Targ;

 %%%KNOY 1st
 Targ.Goals(1) = 4 ;
 Targ.Goals(2) = 54 ;

 %%%OUTSIDE KNOY
 Targ.Goals(3) = 68 ;
 Targ.Goals(4) = 23 ;

 %%%GRANT STREET PARKING
 Targ.Goals(5) = 76 ;
 Targ.Goals(6) = 57 ;

 %%%UNIVERSITY STREET PARKING
 Targ.Goals(7) = 22 ;
 Targ.Goals(8) = 41 ;

 %%%OUTSIDE LWSN
 Targ.Goals(9) = 21 ;
 Targ.Goals(10) = 28 ;

 %%%INSIDE LWSN DOOR
 Targ.Goals(11) = 38 ;
 Targ.Goals(12) = 123 ;
```

```
 %%%LAWSON 1st STAIR
 Targ.Goals(13) = 9;
 Targ.Goals(14) = 101;

 %%%LAWSON 2nd STAIR
 Targ.Goals(15) = 8;
 Targ.Goals(16) = 106;
end
```

initializeTargetArray.m

```
function initializeTargArray()

%Function initializes the states of the targets

global Targ;
 k = 1;

 %%%START
 Targ.State(1,k) = 1 ;    %Start In/Out  (1/0)
 Targ.State(2,k) = 42;    %Start X
 Targ.State(3,k) = 107;   %Start Y
 Targ.State(4,k) = 2 ;    %Start flr   (0/1/2)
 Targ.State(5,k) = 1;     %Start Building (0 - OUTSIDE, 1 - KNOY,
2 - LWSN)

 %%%CURRENT
 Targ.State(6,k) = 1 ;    %Current In/Out  (1/0)
 Targ.State(7,k) = 42;    %Current X
 Targ.State(8,k) = 107;   %Current Y
 Targ.State(9,k) = 2 ;    %Current flr   (0/1/2)
 Targ.State(10,k) = 1;    %Current Building (0 - OUTSIDE, 1 - KNOY,
2 - LWSN)

 %%%INTERMEDIATE
 Targ.State(11,k) = 1;   %Intermediate In/Out  (1/0)
 Targ.State(12,k) = 11;    %Intermediate X
 Targ.State(13,k) = 75;    %Intermediate Y
 Targ.State(14,k) = 2;    %Intermediate flr   (0/1/2)
```

```
 Targ.State(15,k) = 1;    %Intermediate Building (0-OUTSIDE, 1-KNOY,
 2 - LWSN)

 %%%GOAL
 Targ.State(16,k) = 1;    %Goal In/Out  (1/0)
 Targ.State(17,k) = 41;   %Goal X  OLD 41
 Targ.State(18,k) = 61;   %Goal Y  OLD 13
 Targ.State(19,k) = 2;   %Goal flr   (0/1/2)
 Targ.State(20,k) = 2;    %Goal Building (0-OUTSIDE,1-KNOY,2-LWSN)

 %%%MISC
 Targ.State(21,k) = 0 ;   %Step Cost   (1-10)
 Targ.State(22,k) = 0 ;   %Goal State   (0/1)
 Targ.State(23,k) = 0 ;   %GPS      (0/1)
 Targ.State(24,k) = 0 ;   %WiFi     (0/1)
 Targ.State(25,k) = 0 ;   %RFID     (0/1)
 Targ.State(26,k) = 8 ;   %Speed    (1-10)

 Targ.State(27,k) = 0;   %IDX (only for searching nearest nbor)
 Targ.State(28,k) = 0;   %Pause?  (NUMBER TO PAUSE 0 = NONE)

 Targ.State(29,k) = 0;  %Add up step cost, speed and pause cost
end
```

loadFigures.m

```
function loadFigures()

global fig1 fig2 fig3 Nav1 Nav2 Nav3 Nav4 Nav5; % fig3;

load('KNOY1flr.mat');
load('KNOY2flr.mat');
load('PUmap1.mat');
load('LWSN1flr.mat');
load('LWSN2flr.mat');

knoy1 = KNOY1flr;
knoy2 = KNOY2flr;
pumap = PUmap1;
lwsn1 = LWSN1flr;
lwsn2 = LWSN2flr;

clear KNOY1flr KNOY2flr; %PUmap LWSN1flr LWSN2flr;


fig3 = figure('Position',[8 112 695 822]);
fig2 = figure('Position',[703 112 978 822]);
fig1 = figure('Position',[8 112 695 822]);

%%%%%%%%%%%%
%
%  FIGURE 1  - KNOY 1 & 2
%
%%%%%%%%%%%%
```

```
figure(fig1);
subplot(1,2,1), imagesc([0 50], [0 145], knoy1);
yLimits = round(get(gca,'YLim'));
%# Get the y axis limits
yTicks = yLimits(2)-get(gca,'YTick');
%# Get the y axis tick values and subtract them from the upper limit
set(gca,'YTickLabel',num2str(yTicks.'));
%'# Convert the tick values to strings and update the y axis labels
axis image;

hold on;
subplot(1,2,1);
plot(Nav1.wifi(:,1),Nav1.wifi(:,2),'m*', 'MSize', 15)
plot(Nav1.rfid(:,1),Nav1.rfid(:,2),'k*', 'MSize', 15)
subplot(1,2,2), imagesc([0 50],[0 145],knoy2);
yLimits = round(get(gca,'YLim'));
%# Get the y axis limits
yTicks = yLimits(2)-get(gca,'YTick');
 %# Get the y axis tick values and subtract them from the upper limit
set(gca,'YTickLabel',num2str(yTicks.'));
 %'# Convert the tick values to strings and update the y axis labels
axis image;

hold on;
subplot(1,2,2);
plot(Nav2.wifi(:,1),Nav2.wifi(:,2),'m*', 'MSize', 15)
plot(Nav2.rfid(:,1),Nav2.rfid(:,2),'k*', 'MSize', 15)

%%%%%%%%%%%
```

```
%
%  FIGURE 2  - PUMAP
%
%%%%%%%%%%%%
figure(fig2);
imagesc([0 115], [0 82], pumap);
yLimits = round(get(gca,'YLim'));
%# Get the y axis limits
yTicks = yLimits(2)-get(gca,'YTick');
%# Get the y axis tick values and subtract them from the upper limit
set(gca,'YTickLabel',num2str(yTicks.'));
%'# Convert the tick values to strings and update the y axis labels
axis image;

hold on;
plot(Nav3.wifi(:,1),Nav3.wifi(:,2),'m*', 'MSize', 15)

%%%%%%%%%%%%
%
%  FIGURE 3  - LWSN 1 & 2
%
%%%%%%%%%%%%
figure(fig3);
subplot(1,2,1), imagesc([0 45], [0 130], lwsn1);
yLimits = round(get(gca,'YLim'));
%# Get the y axis limits
yTicks = yLimits(2)-get(gca,'YTick');
%# Get the y axis tick values and subtract them from the upper limit
set(gca,'YTickLabel',num2str(yTicks.'));
```

```
%'# Convert the tick values to strings and update the y axis labels
axis image;

hold on;
subplot(1,2,1);
plot(Nav4.wifi(:,1),Nav4.wifi(:,2),'m*', 'MSize', 15)
plot(Nav4.rfid(:,1),Nav4.rfid(:,2),'k*', 'MSize', 15)

subplot(1,2,2), imagesc([0 45],[0 132],lwsn2);
yLimits = round(get(gca,'YLim'));
%# Get the y axis limits
yTicks = yLimits(2)-get(gca,'YTick');
%# Get the y axis tick values and subtract them from the upper limit
set(gca,'YTickLabel',num2str(yTicks.'));
%'# Convert the tick values to strings and update the y axis labels
axis image;

hold on;
subplot(1,2,2);
plot(Nav5.wifi(:,1),Nav5.wifi(:,2),'m*', 'MSize', 15)
plot(Nav5.rfid(:,1),Nav5.rfid(:,2),'k*', 'MSize', 15)
```

updateGPS.m

```
function GPSavailable = updateGPS(x,y,buildings)

global GPSProb;
%Finds signal strength for GPS at given x,y position

a = [x*ones(length(buildings),1),y*ones(length(buildings),1)];
b = (a-buildings).^2;
dist = sqrt(b(:,1) + b(:,2));

buildingNear = min(dist);
probGPSloss =  GPSProb - GPSProb/7*buildingNear;

if probGPSloss <= 0
    GPSavailable = 1;
else
    if rand*100 < probGPSloss
        GPSavailable = 0;
    else
        GPSavailable = 1;
    end
end
```

updateRfid.m

```
function rfidAvailable = updateRfid(x,y,rfid)

%Finds signal strength for RFID at given x,y position
global RfidDist;

TF = isempty(rfid);

if ~TF
    a = [x*ones(length(rfid),1),y*ones(length(rfid),1)];
    b = (a-rfid).^2;
    dist = sqrt(b(:,1) + b(:,2));

    rfidNear = min(dist);

    if rfidNear <= RfidDist
        rfidAvailable = 1;
    else
        rfidAvailable = 0;
    end
else
    rfidAvailable = 0;
end
end
```

updateWifi.m

```
function wifiAvailable = updateWifi(x,y,wifi)

%Finds signal strength for Wifi at given x,y position
global WifiDist;

TF = isempty(wifi);

if ~TF
    a = [x*ones(length(wifi),1),y*ones(length(wifi),1)];
    b = (a-wifi).^2;
    dist = sqrt(b(:,1) + b(:,2));

    wifiNear = min(dist);

    if wifiNear <= WifiDist;
        wifiAvailable = 1;
    else
        wifiAvailable = 0;
    end
else
    wifiAvailable = 0;
end
end
```

updateNavState.m

```
function navState = updateNavState(currPos, InOut, Nav)

%Updates the Nav systems state of an agent
%based on its x,y position

navState = zeros(2,1);
x=currPos(1);
y=currPos(2);

if (InOut == 0)
    navState(1) = updateGPS(x,y, Nav.buildings);
    navState(2) = updateWifi(x,y, Nav.wifi);
else
    navState(1) = updateWifi(x,y, Nav.wifi);
    navState(2) = updateRFID(x,y,Nav.rfid);
end
```

updatePos.m

```
function updatePos(t)

global Targ Path CMap1 CMap2 CMap3 CMap4 CMap5;

%% Check if GOAL is reached
if (Targ.State(6:10) == Targ.State(16:20))
    Targ.State(22) = 1;
    Targ.End = 1;
    %% GOAL not reached
else
    %% Check if INTER GOAL reached
    if (Targ.State(6:10) == Targ.State(11:15))
        %% STILL IN KNOY
        if (Targ.State(10) == 1)
            %% RESET TO KNOY 1st flr
            if (Targ.State(9) == 2)
                %% Set CURR POS
                Targ.State(9) = 1;
                %% SET INTER POS
                Targ.State(12:13) = Targ.Goals(1:2);
                Targ.State(14) = 1;
                 %% MOVE TO OUTSIDE
            else
                Targ.State(6) = 0; %move outside
                Targ.State(7:8) = Targ.Goals(3:4);
                Targ.State(9) = 0;
                Targ.State(10) = 0;
                %% SET INTER POS
```

```
        Targ.State(11) = 0;
        Targ.State(12:13) = Targ.Goals(5:6);
        Targ.State(14) = 0;
        Targ.State(15) = 0;
    end
elseif ((Targ.State(10) == 0) )
    %% MOVE TO 2ND GARAGE
    Targ.State(7:8) = Targ.Goals(5:6);
    Targ.State(10) = 3;
    %% SET INTER POS
    Targ.State(12:13) = Targ.Goals(7:8);
    Targ.State(15) = 3;
 elseif (Targ.State(10) == 3)
    %% MOVE TO OUTSIDE LWSN
    Targ.State(7:8) = Targ.Goals(7:8);
    Targ.State(10) = 4;
    %% SET INTER POS
    Targ.State(12:13) = Targ.Goals(9:10);
    Targ.State(15) = 4;
 elseif(Targ.State(10) == 4)
    %% MOVE TO INSIDE LWSN
    Targ.State(6) = 1; %move inside
    Targ.State(7:8) = Targ.Goals(11:12);
    Targ.State(9) = 1;
    Targ.State(10) = 2;
    %% SET INTER POS
    Targ.State(11) = 1;
    Targ.State(12:13) = Targ.Goals(13:14);
    Targ.State(14) = 1;
```

```
            Targ.State(15) = 2;
        elseif(Targ.State(10) == 2)
            Targ.State(7:8) = Targ.Goals(15:16);
            Targ.State(9) = 2;
            %% SET INTER POS
            Targ.State(12:13) = Targ.State(17:18);
            Targ.State(14) = 1;
        end
    else
        %% CURRENT BLDG KNOY
        if (Targ.State(10) == 1)
            switch (Targ.State(9))
                %% CURRENTLY 1- FLOOR KNOY
                case 1
                    Targ.State(7:8) = CMap1.node((Path(1,t)),1:2);
                    Targ.State(21) = CMap1.node((Path(1,t)),3);
                    %% CURRENTLY 2nd FLOOR KNOY
                case 2
                    Targ.State(7:8) = CMap2.node((Path(1,t)),1:2);
                    Targ.State(21) = CMap2.node((Path(1,t)),3);
            end
            %% CURRENT BLDG OUTSIDE
        elseif ((Targ.State(10) == 0) || (Targ.State(10) == 3)
|| (Targ.State(10) == 4))
            Targ.State(7:8) = CMap3.node((Path(1,t)),1:2);
            Targ.State(21) = CMap3.node((Path(1,t)),3);
        else
            switch (Targ.State(9))
                %% CURRENTLY 1- FLOOR LWSN
```

```
                case 1
                    Targ.State(7:8) = CMap4.node((Path(1,t)),1:2);
                    Targ.State(21) = CMap4.node((Path(1,t)),3);
                    %% CURRENTLY 2nd FLOOR LWSN
                case 2
                    Targ.State(7:8) = CMap5.node((Path(1,t)),1:2);
                    Targ.State(21) = CMap5.node((Path(1,t)),3);
            end
        end
    end
end
end
```

updateTargState.m

```
function  updateTargState(t)

%Function updates the target state based
%on current state and objectives
global Targ Nav1 Nav2 Nav3 Nav4 Nav5;

    updatedState = Targ.State;
    updatedState(21) = 0;
    updatedState(26) = 0;
    updatedState(28) = 0;

        %%%%%
        %%
        %Update nav system
        %%
        %%%%%
    %outside update GPS/WIFI
        if ((Targ.State(10) == 0) || (Targ.State(10) == 3)
|| (Targ.State(10) == 4))

            Nav= Nav3;
            updatedState(23:24) = updateNavState(Targ.State(7:8),
 Targ.State(6), Nav);
            if (updatedState(23) == 1) %if GPS
                updatedState(26)  = 1;
            elseif (updatedState(24) == 1)  %if wifi
                updatedState(26) = 3;
            else
```

```
                updatedState(26) = 10;  %if nothing - slow...
                updatedState(28) = Targ.State(28) + 2;
            end
        elseif (Targ.State(10) == 1)  %inside update WIFI/RFID
            if (Targ.State(9) == 1)
                Nav = Nav1;
            else
                Nav = Nav2;
            end
            updatedState(24:25) = updateNavState(Targ.State(7:8),
Targ.State(6), Nav);
            if (updatedState(24) == 1) %if WiFi
                updatedState(26)  = 5;
            elseif (updatedState(25) == 1)  %if RFID
                updatedState(26) = 7;
            else
                updatedState(26) = 10;  %if nothing - slow...
               updatedState(28) = Targ.State(28) + 2;
            end
        elseif (Targ.State(10) == 2)  %inside update WIFI/RFID
            if (Targ.State(9) == 1)
                Nav = Nav4;
            else
                Nav = Nav5;
            end
            updatedState(24:25) = updateNavState(Targ.State(7:8),
Targ.State(6), Nav);
            if (updatedState(24) == 1) %if WiFi
                updatedState(26)  = 5;
```

```
            elseif (updatedState(25) == 1)  %if RFID
                updatedState(26) = 7;
            else
                updatedState(26) = 10;  %if nothing - slow...
                updatedState(28) = Targ.State(28) + 2;
            end
        end
    Targ.State = updatedState;
end
```

Appendix: C

.5 Patterns of Variables for M_{IV}

Variable Node Count	GPS	Wifi Range	RFID Range	No Runs
Wifi-Rfid				
00-00	60%	5	5	200
00-00	60%	10	5	200
00-00	60%	15	5	200
00-00	80%	5	5	200
00-00	80%	10	5	200
00-00	80%	15	5	200
00-10	60%	5	5	200
00-10	60%	10	5	200
00-10	60%	15	5	200
00-10	80%	5	5	200
00-10	80%	10	5	200
00-10	80%	15	5	200
00-10	60%	5	10	100
00-10	60%	10	15	100
00-20	60%	5	5	200
00-20	60%	10	5	200
00-20	60%	15	5	200
00-20	80%	5	5	200
00-20	80%	10	5	200

00-20	80%	15	5	200
00-20	60%	5	10	100
00-20	60%	10	15	100
00-40	60%	5	5	200
00-40	60%	10	5	200
00-40	60%	15	5	200
00-40	80%	5	5	200
00-40	80%	10	5	200
00-40	80%	15	5	200
00-40	60%	5	10	100
00-40	60%	10	15	100
00-80	60%	5	5	100
00-80	60%	10	5	100
00-80	60%	15	5	100
00-80	80%	5	5	100
00-80	80%	10	5	100
00-80	80%	15	5	100
10-00	60%	5	5	100
10-00	60%	10	5	100
10-00	60%	15	5	100
10-00	80%	5	5	100
10-00	80%	10	5	100
10-00	80%	15	5	100
10-10	60%	5	5	100
10-10	60%	10	5	100
10-10	60%	15	5	100
10-10	80%	5	5	100
10-10	80%	10	5	100

10-10	80%	15	5	100
10-20	60%	5	5	100
10-20	60%	10	5	100
10-20	60%	15	5	100
10-20	80%	5	5	100
10-20	80%	10	5	100
10-20	80%	15	5	100
10-40	60%	5	5	100
10-40	60%	10	5	100
10-40	60%	15	5	100
10-40	80%	5	5	100
10-40	80%	10	5	100
10-40	80%	15	5	100
20-00	60%	5	5	100
20-00	60%	10	5	100
20-00	60%	15	5	100
20-00	80%	5	5	100
20-00	80%	10	5	100
20-00	80%	15	5	100
20-10	60%	5	5	100
20-10	60%	10	5	100
20-10	60%	15	5	100
20-10	80%	5	5	100
20-10	80%	10	5	100
20-10	80%	15	5	100
20-20	60%	5	5	100
20-20	60%	10	5	100
20-20	60%	15	5	100

20-20	80%	5	5	100
20-20	80%	10	5	100
20-20	80%	15	5	100
20-40	60%	5	5	100
20-40	60%	10	5	100
20-40	60%	15	5	100
20-40	80%	5	5	100
20-40	80%	10	5	100
20-40	80%	15	5	100
10-00	60%	5	5	100
10-00	60%	10	5	100
10-00	60%	15	5	100
10-00	80%	5	5	100
10-00	80%	10	5	100
10-00	80%	15	5	100
40-10	60%	5	5	100
40-10	60%	10	5	100
40-10	60%	15	5	100
40-10	80%	5	5	100
40-10	80%	10	5	100
40-10	80%	15	5	100
40-20	60%	5	5	100
40-20	60%	10	5	100
40-20	60%	15	5	100
40-20	80%	5	5	100
40-20	80%	10	5	100
40-20	80%	15	5	100
40-40	60%	5	5	100

40-40	60%	10	5	100
40-40	60%	15	5	100
40-40	80%	5	5	100
40-40	80%	10	5	100
40-40	80%	15	5	100
80-00	60%	5	5	100
80-00	60%	10	5	100
80-00	60%	15	5	100
80-00	80%	5	5	100
80-00	80%	10	5	100
80-00	80%	15	5	100
80-80	60%	5	5	100
80-80	60%	10	5	100
80-80	60%	15	5	100
80-80	80%	5	5	100
80-80	80%	10	5	100
80-80	80%	15	5	100

VITA

VITA

Teresa A. Shanklin

Education

- Ph.D. Computer and Information Technology
  Purdue University, West Lafayette, Indiana
  May, 2012
- M.S. Information Assurance
  Iowa State University, Ames, Iowa
  December, 2006
- B.S. Computer Science, Minor: Mathematics
  Regis University, Denver, Colorado
  May, 2005

Refereed Publications

- T. Shanklin, B. Loulier, E. Matson, Embedded Sensors for Indoor Positioning, IEEE Sensor Applications Symposium 2011. February, 2011, San Antonio, Texas
- K. Lutes, T. Shanklin. So You Want To Teach an iPhone Course, ASEE Global Colloquium on Engineering Education 2011. June, 2011, Las Vegas, Nevada
- T. Shanklin, B. Loulier, E. Matson and E. Dietz, An Applied Multi-agent System Simulation for Path-Planning on a Mobile Device, 2012 Springer 10th International Conference on Practical Applications of Agents and Multiagent Systems, PAAMS 2012, Salamanca, Spain

Submitted

- T. Shanklin, E. Matson, C. Tytler, G. Lo, J. Altchuler, A Multi-agent System-of-Systems Model to Navigate a Complex Campus Environment, SummerSim 2012, Genoa, Italy
- T. Shanklin, B. Marshall, E. Matson Data Fusion from Embedded Sensors on Smart Phones for Improved Localization, ACM SIGMOBILE 18th Annual International Conference on Mobile Computing and Networking 2012, MOBICOM 2012, Istanbul, Turkey
- T. Shanklin, B. Marshall, E. Matson Agent-Based Model and Simulation for Multi-Sensor Navigation, 2012 The Journal of Ambient Intelligence and Smart Environments, Technical Journal

Teaching

- CIT 155: Introduction to Object Oriented Programming. F08, S09, F09 (teaching assistant). C#
- CIT 295: Object-Oriented Programming. F11 (teaching assistant). JAVA
- CIT 315: Systems Programming. S12 (teaching assistant). C

Awards

- Grad Cohort Scholarship Recipient 2010.
- Grace Hopper Scholarship Recipient 2010.
- Purdue Research Fellowship Award. 2010-2011.

Professional Experience

- Graduate Research Assistant. S09.
  Professor Rick Mislan
  Northrup Grumman Cybersecurity Research Consortium
  Research into fast forensics, improving the speed and fidelity of forensics in the field on cell phones, PDAs and similar devices.

- Purdue Research Fellowship Award. F10-S11. (research assistant)
  Professor Brandeis Marshall

  Project, Seeing iPhone

  Research into viability of using embedded sensors on Apple iPhone to provide indoor localization and navigation for visually impaired users.

- Owner, Shanklin Computers Services and GTM Process, Inc. Fairbanks, AK
  2001 - March, 2010

  Write customized software program for contract clients (State of Alaska/ Lockheed Martin)

  Provide various computer security services for contract clients: application & network security.

  Maintain company security including disaster recovery and data integrity.

  Create custom application for business including database.

- Systems Administrator, SMI/ NOAA/ Lockheed Martin, Fairbanks, AK
  2001 - 2005

  Assessed computer and network security vulnerabilities, and reported findings to senior management with recommendations for corrective action.

  Collaborated on security issues with System Security Officer at NOAA headquarters in Suitland, MD.

  Configured and managed user access, passwords, permissions, and restrictions on Windows local and wide area networks.

  Configured system backups, verified backup data integrity and initiated backup/recovery strategy.

  Installed, configured and maintained a variety of security equipment, including Cisco routers, firewalls and VPNs.

  Authored local station security procedures, and ensured compliance and reporting for NOAA and Department of Commerce policies.

- Lead Systems Administrator, Aschenbrenner Law Offices, Fairbanks, AK

1994 - 2000

Development of custom software program, providing automated document creation through Access database.

Improved program utilized for State of Alaska reporting purposes by modifying and improving existing custom code to meet changing needs and requirements.

Upgraded and enhanced network; transitioned from peer-to-peer COAX, to Cat3 cabling with Novell 2.6 server then 3.1 and Windows 3.1. Migrated system to Windows NT, 2000, 2003 network with Cat5.

Setup and configured Cisco routers and Cisco PIX between Fairbanks and Anchorage offices using VPN technology.

Boosted efficiency and reduced errors by creating new legal database to automate creation of numerous forms and document.

Responsible for all system security, including disaster recovery.

Affiliations

- Institute of Electrical and Electronics Engineers (IEEE), Student Member
- Association for Computing Machinery (ACM), Student Member
- Information Systems Audit and Control Association (ISACA), Member
- International Information Systems Security Certification Consortium, Inc. (ISC)2, Member
- Phi Kappa Phi Honor Society, Member
- Center for Education and Research in Information Assurance and Security (CERIAS)
- Purdue M2M Lab

Industry Certifications

- CISA - Certified Information Systems Auditor

- CISSP - ISC Systems Security Professional
- GSEC-GIAC Security Essentials
- MCSE - Microsoft Systems Engineer, NT
- GCIH-GIAC Incident Handler
- MCP - Microsoft Professional, W2K
- GREM-GIAC Reverse Engineer Malware
- CCNA-Cisco Network Associate
- GISP-GIAC Security Professional
- CompTIA A+
- GCFW-GIAC Firewall Analysis
- MCT Microsoft Trainer
- GCFA-GIAC Forensic Analyst

Technical Industry Training

- Cisco Secure IDS
- Linux Administration
- Cisco Secure PIX Firewall Advanced
- Cisco Router Programming
- Sans Firewalls, Perimeter & VPNs
- MCT-Train the Trainer
- Sans Hacker Techniques & Incident Handling
- Cisco Interconnecting Network Devices
- Sans +STM Traning Program for CISSP Exam
- Cisco Interconnecting Network Devices
- Sans Reverse Engineering Malware

- Windows Internetworking TCP/IP
- Sans System Forensics
- Windows NT Enterprise Technology
- SANS Training for CISA Exam
- Copper & Fiber Cabling
- Black Hat Malware Analysis
- Sans Drive and Data Recovery Forensics
- Advance Memory Forensic in Incident Response