PURDUE
UNIVERSITY

Purdue University West Lafayette, Indiana

College of Technology

Using smartphones for indoor navigation

In partial fulfillment of the requirements for the
Degree of Master of Science in Technology

A Directed Project Report

By

Benjamin Loulier

11/28/11

Committee Member	Approval Signature	Date
Eric T. Matson, Chair	_____	_____
Brandeis Marshall	_____	_____
James E. Dietz	_____	_____

TABLE OF CONTENTS

Page

Page

Page

1. INTRODUCTION

This chapter introduces the problem with an overview of why it is significant, and identifies the scope and the goals of the study. A few definitions of generic concepts that are used widely in this report are also provided, before providing a generic frame of the study.

1.1 Statement of the problem

Getting an accurate positioning is a central issue in many applications ranging from robotics, transportation to the everyday life of people. One system is currently widely adopted to answer this problem: The Global Positioning System aka GPS. The GPS, created in 1973 by the US Department of Defense (USDOD) is currently run using a set of geostationary satellites. Each satellite contains an atomic clock and broadcast a radio wave containing a time signal. On the ground a GPS receiver reads the time contained in the signal and compare it to the time of reception, this tiny time difference along with an accurate model of the propagation of the radio wave throughout the different layers of the atmosphere allows to compute the distance between the receiver and the satellite. The same measure is done for different satellites and then, using triangulation an estimation of the position can be computed.

Although very accurate, one flaw of the GPS system is that it is not adapted for indoor environment. As the radio waves travel a long way the attenuation is very important and any physical obstacle can greatly deteriorate the quality of the signal. As of today the problem of indoor positioning is addressed mainly in three different maners:

- GPS like approaches where other external references are used instead of geostationary satellites. The positioning using wifi hotspots and cell towers is a

good example. Similar methods have been developed doing triangulation using RFID tags.

- Landmark-based methods consists in having a grid of devices regularly positioned in your indoor space. The industry widely uses these methods, using for instance solenoids embedded in the floor that the mobile robots can follow.

- Inertial Navigation is the last method. The concept of inertial navigation is very simple, knowing the initial position of a moving object, its speed, and the time during which it has been moving we can estimate the position of an object. Inertial Navigation is usually achieved using accelerometers and gyroscopes.

While the cost of a GPS sensor is currently as low as $14 in the sparkfun catalog, the gyroscopes and accelerometers used for this research (Ojeda & Borenstein, 2006) about indoor positioning for pedestrians cost around $1200. Currently, from a general public prospective, the cost of inertial navigation is much higher than the price of navigation using GPS. However, a first identification of the issue acknowledges that there is a large field of applications of indoor positioning for the general public, an obvious one is point to point navigation in large buildings such as malls, hospitals or campuses. Disabled people could also benefit from a cost effective solution for indoor positing.

Another aspect of the problem is the navigation software part, we have to develop a solution that is flexible enough to support different types of positioning sources and modular enough to be implement on multiple platforms using one or multiple devices.

1.2 Research question

The question of the study is:

How can we develop a solution for indoor navigation using new generation smartphones ?

1.3 Scope

To build a system which is both cost effective and easily available for the general public, we chose to focus our study on the new generation smartphones which are now owned by many people.

The study is composed of three parts:

1. The first part is the study of a Inertial Navigation method for pedestrians, the different challenges it poses and a study of feasibility concerning the implementation of this method on an iPhone 4.

2. In the second part, we develop a landmark-based method using RFID tags containing latitude/longitude information and read by a Nexus-S Samsung phone.

3. The third part focuses on the design of a theoretical multi agent framework for the development of navigation softwares. We also develop as a proof of concept a prototype implementing this framework on an iPhone 4.

The two first part of the project present two positioning methods that can be used indoor, and the last part tackle the navigation part of the research question, this is how the three parts are related to the main topic of the directed project which is "Indoor navigation for new generation smartphones". The final goal would be to integrate these three projects together on a smartphone to make a complete system for indoor navigation, however this is out of the scope of this directed project.

1.4 Significance of the problem

Positioning is crucial in many applications, the GPS revolution in 1973 brought the possibility to theoretically locate any objects on the world. This new technology impacted greatly the world of IT both in the military and in the civil, where first sailors and pilots started using GPS finally followed by regular people in their car. The first application of GPS is navigation but with the emergence of social media and interactive advertisement new uses where position is central started to appear

approximatively three years ago. For instance Facebook allows you to check-in in different places given your position, new search engines recommend restaurants based on your location and a different add will be displayed on your screen given the streets you are walking on. Those applications cannot work reliably indoor right now because GPS is not reliable inside. Therefore, given all the applications indoor positioning could have just by transposing the applications using the GPS, solutions for indoor positioning are actively researched and furthermore represent a big challenge for the industry.

Until the 1990's technology has been a blocking factor for research about indoor positioning. A first attempt to design an indoor positioning system was made by Want, Hopper, Falcão, and Gibbons (1992) in Xerox Research Park in Palo Alto. This basic system uses the security badges of employees to locate them in the building.

Research about indoor positioning systems began to be very popular when radio wave emitting devices such as GSM towers and Wi-Fi access points became widespread. Drane, Macnaughtan, and Scott (1998) started doing research about using GSM cell towers to locate cell phone, this research was driven by the new FCC requirements for 2001 stating that every 911 call can be located precisely. Compared to the GPS radio waves, RFID and Wi-Fi signals are less attenuated by obstacles and so are a better source for indoor positionning. The multiple path problem remains but we won't talk about it as it is not the focus of our study and as methods as fingerprinting don't suffer from this problem (Kaemarungsi & Krishnamurthy, 2004). Since then a lot of research has been done to design indoor positioning using Wi-Fi and more recently RFID. These researches put the spotlight on indoor positing and new methods such as the use of Inertial Navigation were also experimented. Curently, indoor positioning is still a very active research area and with the new wave of smartphone we started to see application appearing on the market, companies such as Nokia started to make great effort to develop indoor positioning solutions Kalliola (2008).

To conclude, as there are numerous potential application and technology is getting very mature, indoor positioning can be qualified as a hot topic in actual research.

1.5 Definitions

Smartphone. According to "Oxford English Dictionnary" (2010) a smartphone is "any of various telephones enhanced with computer technology; (now) spec. a type of mobile phone which incorporates the functions of a palmtop computer, personal digital assistant, or similar device".

Inertial Navigation. Inertial Navigation is defined in Woodman (2007) as "a self-contained navigation technique in which measurements provided by accelerometers and gyroscopes are used to track the position and orientation of an object relative to a known starting point, orientation and velocity" (p.5).

RFID. RFID as defined in Landt (2005) is "a term coined for short-range radio technology used to communicate mainly digital information between a stationary location and a movable object or between movable objects" (p. 1). A typical RFID system is composed of a tag that contains the data and a reader which is able to retrieve the data contained in the tag. Typically,"the reader sends an unmodulated signal to the tag" (p. 1) and according to the information it contains, "the signal reflected from the tag is modulated with this coded information" (p.1).

Agent. According to Weiss (1999) "there is no universally accepted definition of the term agent, and indeed there is a good deal of ongoing debate and controversy on this very subject" (p.28). However Wooldridge and Jennings (1995) defines two types of agency. A weak agent is a computer system that is autonomous, has some social abilities which means it can communicate with other agents and possibly humans, is reactive which means it perceives and reacts to its environment, and lastly pro-activeness which means that they are able to ex-

hibit goal-directed behaviour. A strong agent is an agent which "in addition to having the properties identified above, is either conceptualised or implemented using concepts that are more usually applied to humans" (p. 5).

1.6 Assumptions

The assumptions for this study are:

- All the softwares and methods are applied on devices meeting the minimum sensor and power requirements.
- The pedestrian positioning system using Inertial Navigation assumes a consistent walking attitude.
- The indoor environment contains even surfaces, we won't consider irregular terrain because it might generate too much noise for the sensors.
- For this study we only consider 2D environments.

1.7 Limitations

The limitations for this study are:

- The softwares developed are associated to a specific device and a specific system embedded on this device.
- Only the sensors embedded in the smartphone will be used.
- For the part concerning the study of a landmark-based method using RFID tags for positioning, we will only use RFID tags as containers for the latitude and longitude of the place they are located

1.8 Delimitations

The delimitations for this study are:

- No integrated system will be developed, we will only consider parts separately (positioning and navigation).

1.9 Summary

This chapter presents the subject of the directed study, what kind of problem the study is trying to address, what is the scope of the study and the significance of this problem in the scientific community. We then introduce definitions of concepts that are used in the rest of the report and the assumptions, limitations and delimitations of the study. The next chapter presents the literature review for this study.

2. LITERATURE REVIEW

As stated before, this study is trying to answer the following question: *How can we develop a solution for indoor navigation using new generation smartphones ?* and is organized in three different parts: Inertial Navigation for pedestrians, landmark-based method using RFID tags for positioning, multi agent framework for navigation software.

In this chapter we present the literature review associate to every part of the study.

2.1 Inertial Navigation for pedestrians

An accelerometer and digital compass from a mobile phone are used by Constandache, Choudhury, and Rhee (2010) for localization. The authors use these sensors to measure the walking speed and orientation of the user. They take these data points from the mobile device's sensors as they are more energy efficient than GPS and Wi-Fi based systems. It is not as accurate however. The main purpose of this approach is to conserve energy and make use of methods that are infrastructure independent. The authors propose their method as a complement to existing methods that are more accurate, but energy draining.

The accelerometer data is used as input for a pedometer application by Dekel and Schiller (2010). The authors achieve fairly accurate results using un-augmented Smartphone's, but require the approximate measure of a users' stride to function effectively.

Hynes, Wang, and Kilmartin (2009) use an embedded accelerometer to record and analyze gait. The study uses a low pass filter to detect periods in the data. This allows the researchers to determine between two states dubbed walking and non-walking. Liu

et al. (2010) also use the embedded accelerometer to determine the periodic pattern of each step. This permits the authors to assist the WLAN positioning by providing user context detection, i.e. is the pedestrian static, walking slowly or walking fast. The contribution of the authors is three fold: a Weinbull function is used to represent the wireless signal strength distribution over time; accelerometer data is added to improve accuracy; and a hidden Markov Model particle filter is used to find the indoor position from both inputs. The drawback is this strategy requires a database-training phase for the wireless positioning.

Hsu and Yu (2009) present a theoretical and simulated version of accelerometer based positioning. There is a linearly increasing error that is directly proportional to the elapsed time.

Accelerometer data is used to classify the user activity in the method proposed by Parnandi et al. (2010). This is combined with the last known GPS coordinate when the user came indoors to provide a coarse indoor location of the user. The goal of the system is to provide localization information without external infrastructure. The following method is employed: the GPS coordinates for the last known outdoor location are logged. Once the user moves inside, accelerometer data is logged to a file at specific intervals. Once the user stops moving, the changed states causes 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 users activities: standing still; walking; going up or down stairs; 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.

Ofstad, Nicholas, Szcodronski, and Choudhury (2008) use accelerometer readings from a mobile phone to determine a users current location on Google Maps as well as the context of their activity. In this system, the time interval for sensor recording is set to one second. At each one minute mark, a layer filter classifies and records the

users current activity, i.e. sitting or standing. This was a preliminary work that the authors expect to expand into a fuller featured system.

2.2 Landmark-based method using RFID tags for positioning

Want (2006) in his paper gives an overview of the RFID technology. The author classify the RFID technology: passive and active. A passive tag is not powered and only emits when some current is inducted in the tag by a reader, an active tag contains a source of power and therefore emits constantly even without the presence of a reader. The author also makes a difference between Near-Field and Far-Field RFID, this distinction is done according to the range the tag can be detected by a reader. Most of the active RFID tags can be used as Far-Field, it is different with the passive tags as the further you get from the tag the higher is the energy you need to emit to induce power into the passive tag.

A first method to achieve positioning using RFID tags is using the tags or readers as landmarks.

Yelamarthi, Haas, Nielsen, and Mothersell (2010) describes a mobile robot for visually impaired people which uses while indoor a combination of infrared sensors, ultrasonic sensors and RFID tags to estimate the position of the disabled person and provide him with haptic and vocal feedback so that he stays on track. The RFID tags are used as indoor landmarks, each RFID tag contains its location, when the robot get close to the tag it detect the presence of the tag and reads its content to infer the current position.

Silva, Paralta, Caldeirinha, Rodrigues, and Serodio (2009) presents another method to locate objects and users using RFID. Compared to the previous method the implementation is reversed, each user or object to locate is assigned a tag and readers are scattered in the different rooms of the building. The readers are connected to a network and transmit information to a server running a specific middleware TraceMe developed by the authors. Every time a reader detects a tag an information is sent

to the server and as the location of the reader is known the position of the user is estimated. This paper also presents an interesting method using the RSSI and the angle of the RFID signal to create a network of RFID readers able to self calibrate using only a few primary references.

RFID can also be used to position users and objects using triangulation.

Fu and Retscher (2009) presents the use of RFID tags as a media to achieve positioning using trilateration. The system retrieves the strength of the signal (RSSI) and given the attenuation computes a distance between the reader and the RFID tag, this information gives a circle in which the user could be located. Using multiple tags, the position of the user is computed as the intersection of the circles corresponding to all the tags. The accuracy achieved for the positioning is close to two meters. Joho, Plagemann, and Burgard (2009) propose a refinement of the trilateration method by combining it with a statistical model of the radio wave propagation developed using statistical machine learning. With this combined system they manage to achieve an accuracy of 29cm for the estimation of the position of an RFID tag.

2.3 Multi agent framework for navigation software

Muñoz Salinas, Eugenio, Garcia-Silvente, and Gómez (2005) describes a layered hybrid architecture to handle mobile robots navigation. The system is composed of a deliberative layer where the planner decides which is the best plan to follow to get to the destination given mapping informations; An execution and Monitoring layer where the plan is transformed in a sequence of actions given the skills of the robot, the execution of the plan is then monitored; A control layer which contains agents in charge of gathering sensory inputs from the environment, the agents uses fuzzy logic to handle the errors of the sensors.

Li, Christensen, Oreback, and Chen (2004) presents an open source framework to design control systems for mobile robots evolving in a dynamic unknown environment. This framework use a set of standardized communication patterns, CORBA

as a communications channel. This architecture is composed of different specialized components to handle the sensory inputs, the path planning and the user interface.

The multi-agent approach for navigation has been used in the context of applications designed for impaired people. Falco et al. (2010) designed a multi-agent system for the elderly and people with disabilities. This system introduces an architecture with five specialized agents in charge of path planning, path building, orientating the user, Tracking the position of the user, Locating the user on the map. Morère and Pruski (2004), use a multi-agent approach to create an intelligent wheelchair able to select the type of control it should use depending on the environment to allow the user to navigate more easily, precisely, and safely.

In Deloach, Wood, and Sparkman (2001) DeLoach (2006) and DeLoach, Oyenan, and Matson (2008) the Multi agent System Engineering (MaSE) methodology is developed as a further abstraction of the object-oriented paradigm. The methodology is broken into seven steps along a logical progression: capturing goals; applying use cases; refining roles; creating agent classes; constructing conversations; assembling agent classes; and system design.

Yu (2001) and Kelemen (2003) describes Agent Orientation, a similar paradigm as MaSE for software engineering. To provide more flexibility and reusability the Agent Orientation paradigm proposes to model softwares as set of autonomous agents communicating together.

3. METHODOLOGY

As stated before, this study is trying to answer the following question: *How can we develop a solution for indoor navigation using new generation smartphones ?* and is organized in three parts:

- Inertial navigation for pedestrians
- Landmark-based method using RFID tags for positioning
- Multi agent framework for navigation software

We first study two different methods of positioning that can be used in an indoor environment and then present a framework for the development of a navigation software. These three parts are studied independently but the final foal is that they are integrated to create a complete indoor navigation system on smartphone, however the integration is out of the scope of this directed project.

For each part of the study we will present the different methods employed both for the data collection and evaluation.

3.1 Inertial navigation for pedestrians

3.1.1 Specifications

We will use an iPhone 4 for this part of the study. The data from the sensors of the iPhone will be gathered using an embedded software realized in Objective-C,

C. A prior study from the author revealed that the current version of the iPhone operating system (OS) iOS4 offer several methods to access the sensor data.

3.1.2 Data collection & Methodology

For this part of the study we are dealing with purely quantitative data: raw data from iPhone sensors, 2D position information after the Inertial Navigation operation. The data will be collected and stored in a csv file so that we can plot the estimated position of a test user. Following is the methodology used to develop the study:

- Model theoretically inertial navigation applied to pedestrians. This step is purely theoretical and does not involve any data collection.
- Retrieve iPhone sensors data that are necessary to compute the position using the inertial navigation principle. To avoid any errors while retrieving the data we will do some test to avoid any inconsistencies in the sensor data, we will make sure that when the iPhone stands still we have the readouts we are supposed to have taking into account the fact that every sensor also generates noise. To avoid introducing a bias due to the device in our study we will use two iPhones 4.
- Create and calibrate a filter to achieve inertial navigation. The calibration of the filter is critical and very dependent on the sensors characteristics, the calibration method is mostly empirical and to do so we will use a trial and error method.
- Make a testing campaign to evaluate the accuracy of Inertial Navigation. To avoid bias we will do the test campaign using different person and asking them to walk at different paces, to ensure repeatability we will ask every person to do multiple runs and to ensure reproducibility we won't do all the tests at the same time.

3.1.3 Evaluation

The precision measure we will use to evaluate the quality of our system is the distance between the real position of the user and the position computed by our system. Computing the average over all the tests we do will give us a good estimation of how our system performs.
We performed the tests with three different users, each one of them walking three times in straight line for four meters.

3.2 Landmark-based method using RFID tags for positioning

3.2.1 Specifications

We will use a Samsung Nexus-S smartphone for this study because it has an RFID reader, we will also use passive RFID tags. The software used to read the content of the RFID tags and translate them into a position on the map is going to be realized in Java.

3.2.2 Data collection & Methodology

To develop this research we will follow the following plan:

- Get position of reference for the RFID tags. To do this we will use a map of the building we will use for our tests and compute the lat/long of the RFID tag accordingly, the measure on the map will have to be very precise because we will be dealing with very small differences in latitude and longitude.
- Develop a software on the Nexus-S allowing us to overlay any map of a building over the regular map view.
- Use this software along with the RFID tags to show the position of the user on a map of the building every time a new tag is scanned.

3.2.3 Evaluation

The tool we develop for this part of the study is a proof of concept. The main criteria for evaluation is wether the tool work or not. The two main task the tool has to achieve are :

- Display maps adapted to an indoor environment.
- Position the user on this map using the RFID tags he scans with his phone.

3.3 Multi agent framework for navigation software

3.3.1 Specifications

We will use an iPhone 4 to develop an implementation of our multi agent model, the software will be realized in Objective-C and C.

3.3.2 Methodology

We will use the following plan to develop the research:

- Development of a theoretical multi agent framework for navigation applications. This part is a model research where we investigate the best way to conceive a navigation application using a multi agent paradigm.
- Development of a software implementing this multi agent framework. In this part we create a prototype on a smartphone which implements the multi agent framework developed in first place.

3.3.3 Evaluation

This part of the study is focused on the design of a model. Two validations are important for a model: the internal validation ensures that the logic of the model is

consistent, and the external validation ensures that the model can be used in multiple implementation contexts.

The model will be validated internally using the software we develop as a proof of concept, if we manage to implement the model on the iPhone and have a functioning software it means that our model is internally valid. To achieve external validity for the model we will focus during the design of the model on staying very general and don't use any specificities of any platform where the multi agent model could be implemented.

4. INERTIAL NAVIGATION FOR PEDESTRIANS

This chapter is extracted from a paper published to the Sensor Application Symposium 2011 (Shanklin, Loulier, & Matson, 2011).

4.1 Introduction

Developing effective indoor navigation systems for a variety of purposes is a research focus that has been studied often in recent years. The goal of this study is to develop a useful indoor positioning system using a low cost, publicly available device. This study focuses on the fourth generation iPhone as it contains embedded sensors: accelerometer; gyroscope and magnetometer. Although the theory of inertial navigation systems dates back thousands of years, using inexpensive sensors in a public platform is a novel and difficult task.

4.2 Design

In this section we present the basic physics behind inertial navigation and the characteristics, the sensors we are going to use in this study and introduce the drift issue we will face in this study.

4.2.1 Projection

We want to give the movement of the user relative to the earth referential. To do this we need to express the accelerations in the earth referential. The gyroscopes allow us to measure the attitude of the phone. Knowing this attitude we can project the acceleration vectors - expressed in the referential of the phone - on the earth referential.

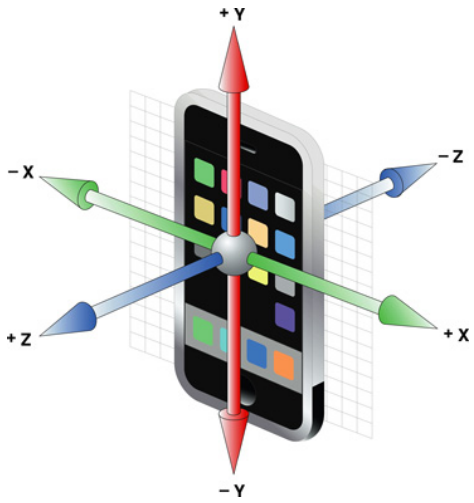

Figure 4.1. This figure shows the coordinate system attached to the iPhone in which the accelerations are expressed.

4.2.2 Integration

The theory behind an inertial navigation system is basic. The accelerometers and gyroscopes allow us to get the acceleration in a referential tied to the earth. Using the approximation that this referential is Galilean, we use two integration steps to get the position from the acceleration. Integrating the acceleration we can get the speed as seen in Equation (4.1). Only equations projected along the x-axis are shown for brevity.

$$\int_0^t a_x(t)\,\mathrm{d}t = v_x(t) - v_x(0) \tag{4.1}$$

And integrating the speed we can get the position shown in Equation (4.2).

$$\int_0^t v_x(t),\mathrm{d}t = x(t) - x(0) \tag{4.2}$$

4.2.3 Measurement issues

4.2.3.1. Drift due to integration

No sensors are perfect; despite filtering there is always an error. Integrating a measure will result in a linear drift after the first integration as seen in Eq: (4.4) and a squared drift after the second shown in Eq: (4.5). Let A_x be the value of acceleration measured along the x-axis, a_x the real value of the acceleration and ϵ the error.

$$A_x(t) = a_x(t) + \epsilon \tag{4.3}$$

$$\int A_x(t), \mathrm{d}t = \int (a_x(t) + \epsilon), \mathrm{d}t = v_x(t) + \epsilon t + \alpha \tag{4.4}$$

To simplify the equation let's say the integration constant $\alpha = 0$, we can observe the linear drift ϵt due to our error in the measure of the acceleration. This time we observe a squared drift ϵt^2

$$\iint A_x(t), \mathrm{d}t = \int (v_x(t) + \epsilon t), \mathrm{d}t = x(t) + \frac{1}{2}\epsilon t^2 + \beta \tag{4.5}$$

The following figure 4.2.3.1 presents the difference between a model of the speed curve for somebody walking and a curve that would be affected by a drift.

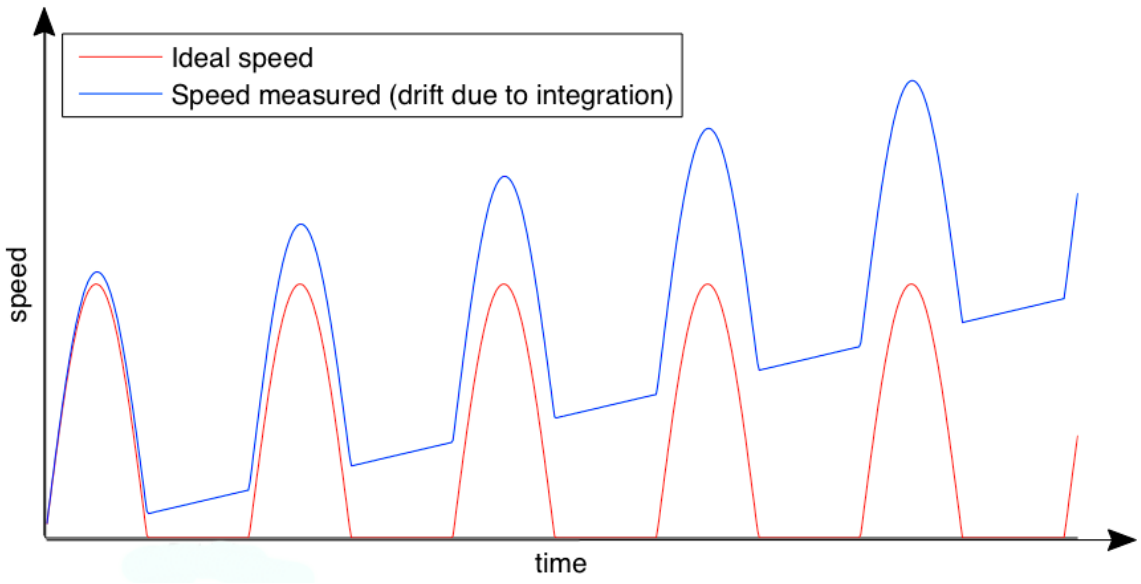

Figure 4.2. Illustration of the drift

4.2.4 Embedded sensors

4.2.4.1. Accelerometer

The iPhone has an ultra-low power digital three-axis accelerometer by STMicroelectronics (LIS331DLH). According to the datasheet of the constructor this accelerometer can be configured to measure acceleration data between $\pm 2g/\pm 4g/\pm 8g$. Our experiments shown in the following table indicates that Apple configures it in a $\pm 2g$ range.

Range (g)	±2
Acceleration noise density ($\mu g/\sqrt{Hz}$)	218
Bandwidth (Hz)	25

Table 4.1
Accelerometer main characteristics

4.2.4.2. Gyroscope

A three-axis gyroscope by STMicroelectronics (ST) is included in the iPhone 4 (L3G4200D). A gyroscope is used to determine the rate an object rotates, integrating these data over time allows us to get the pitch, yaw and roll of the phone - again because we have an integration step this measure suffers from drift.
According to the datasheet of the constructor this accelerometer can be configured to measure acceleration data between $\pm 250 dps / \pm 500 dps / \pm 2000 dps$. Our experiments shown in the following table indicates that Apple configures it in the $\pm 250 dps$ range.

Range (dps)	±250
Acceleration noise density (dps)	0.03
Bandwidth (Hz)	40

Table 4.2
Gyroscope main characteristics

The drift observed is low - almost nonexistent for the pitch and roll when the iPhone is standing still. The Apple API we use to access the roll and pitch of the iPhone already implements an algorithm to remove the drift using the direction of the gravity (given, thanks to the accelerometer). The drift is a little more important for

the yaw but again, in our experiment having the iPhone standing still, we observed a drift of less than one degree over ten minutes.

Managing to get a high quality Inertial Measurement Unit (IMU) from these sensors is going to be quite challenging because although the new sensors embedded in the iPhone 4 are more accurate than the one in the previous versions they are still small and cheap sensors which cause the resulting signals to be noisy.

4.2.5 APIs available in iPhone OS used to gather the sensors data

In iPhone OS 4 Apple introduced a new framework called CoreMotion (2010) .This framework allows us to access both low level information, like the rotation rates of the gyroscopes or the raw acceleration data, but also more high level data like the user acceleration (gravity is filtered out) or the pitch, roll and yaw of the device.

4.3 Implementation

Only the sensors embedded on the iPhone are used.

In the first system we used, the measure and filtering were embedded on the iPhone (a part of the results presented were acquired using this approach).

To get more flexibility and make easier to change the filtering parameters this data is now sent via UDP packets over the Wi-Fi network to a computer running LabView. We are using its provided processing functionalities to determine the most appropriate parameters for filtering and integration.

4.3.1 Attachment of the device to the body

The body is - fortunately - not completely rigid and has many degrees of freedom (DOF). This presents a real challenge for our system, as many DOF mean a lot of unwanted movement interfering with our measure such as trembling or vibrations. Initially we attached the iPhone to the foot, a body part with a consistent and stable

motion when walking which allowed us to use the Zero Velocity Update principle (Skog, Händel, Nilsson, & Rantakokko, 2010) to remove the drift due to the integration of the speed.

4.3.1.1. Zero Velocity Update principle

The basis of this principle is that as we walk, one foot moves and then stands still. So if we can detect the end of a step and the beginning of the next one we can assume that during this period the speed of the foot is zero. This allows us to recalibrate the speed, dramatically decreasing the effect of the drift.
We are considering a solution to detect a step :

- Detecting the minimum of the acceleration norm: $|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. A simple version of this solution using a simple threshold is shown in Table 4.3.

if $\|a\|$ is $<$ threshold
then.. reset Initialization variable to true

Figure 4.3. Re-initialization to account for drift

- Detecting the minimum of the gyros rotation rate.

The following figure presents a model of the speed of the foot for a pedestrian walking and the same curve affected by the drift, it also presents the steps where you can apply Zero Velocity Update.

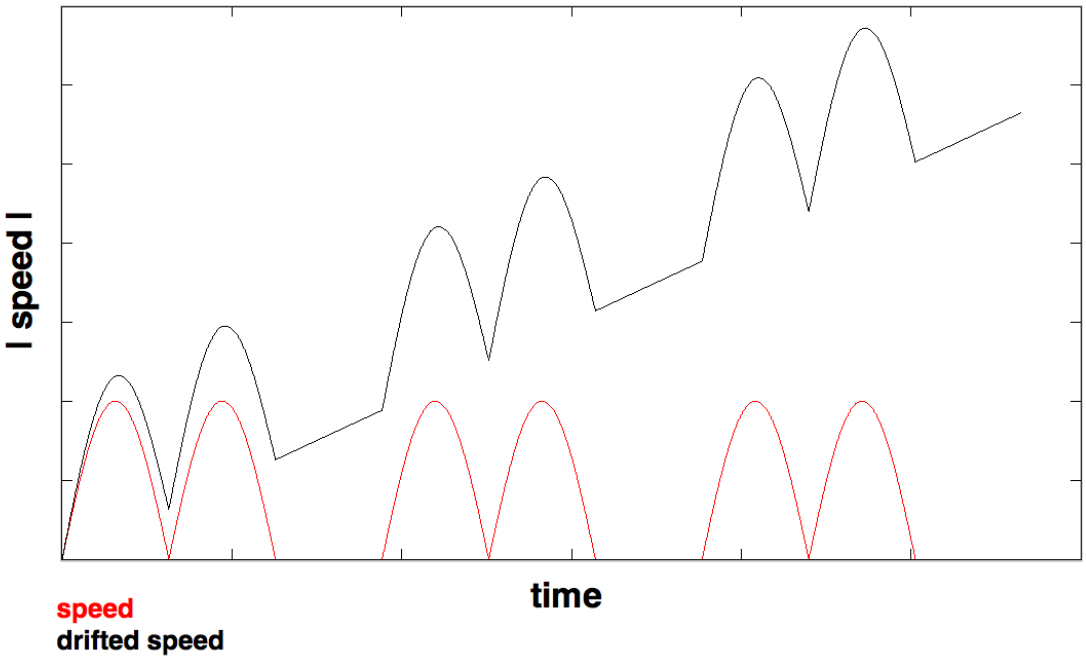

Figure 4.4. Zero Velocity Update

Having the phone strapped to the foot is not a practical solution, but we are trying to identify some typical behavior of the movement when the iPhone is in the hand or in a pocket which will allow us to use a method similar to the Zero Velocity Update with the foot movement. But as the movement of the body is less consistent than the movement of the foot while walking, it is difficult.

4.3.2 Overview of the system

An overview of the system we use for this research is shown in Fig: 4.5. Steps of the system are also detailed.

4.3.2.1. Low pass filtering

We get from the iPhone a measure of acceleration where the constant due to gravity has been removed. The acceleration due to the motion is a low frequency signal, the noise being an high frequency we isolate the signal from the noise using a low pass filter. A low pass filter is an electronic filter removing from the signal frequencies higher than a certain cutoff frequency.

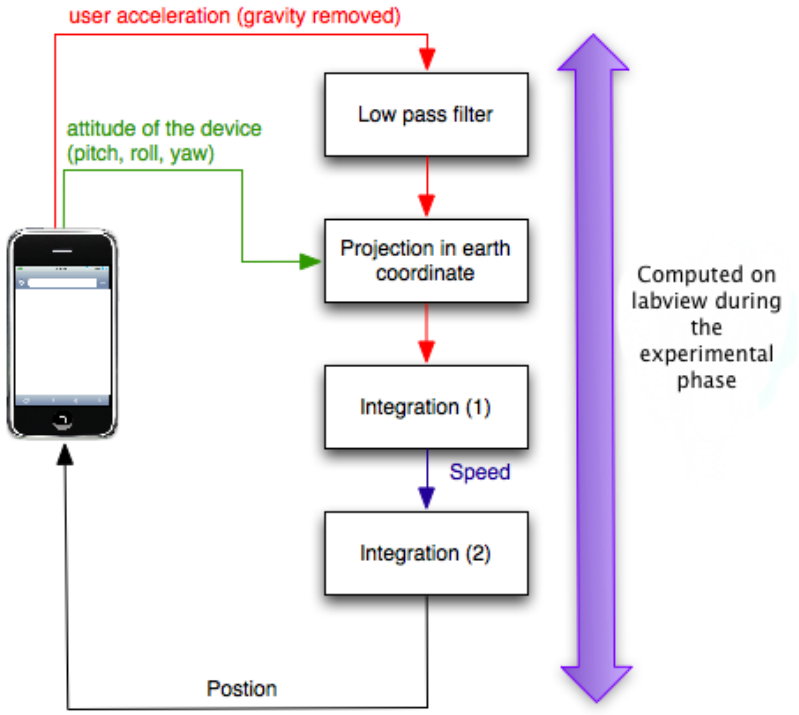

Figure 4.5. Overview of our system.

4.3.2.2. Projection

To do the projection we use the following matrix which is the inverse of a rotation matrix with the three Euler angles of the phone ($\psi = yaw$, $\theta = pitch$, $\phi = roll$).

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

Figure 4.6. Projection matrix, cosines is abbreviated whit c and sinus with s

4.3.2.3. Integration (1)

The first integration presents the particularity that we are going to use the Zero Velocity Update principle to remove or reduce the drift.

4.3.2.4. Integration (2)

The second integration will provide pertinent information regarding the position of the user. This will be tied to the path-planning portion of the project, and results shown for this are broad and general.

4.3.3 Labview Virtual Instrument

As stated before we used Labview to do all the filtering and the processing of the initial gyroscopes and accelerometers signals to obtain the position. The sensors data is sent from the iPhone to the iPad over UDP, the two following figures 4.7 4.8 present our Virtual Instrument in Labview and breaks down all the functionality's embedded

into it, as you can see you can find all the steps described in the subsection Overview of the system 4.3.2.

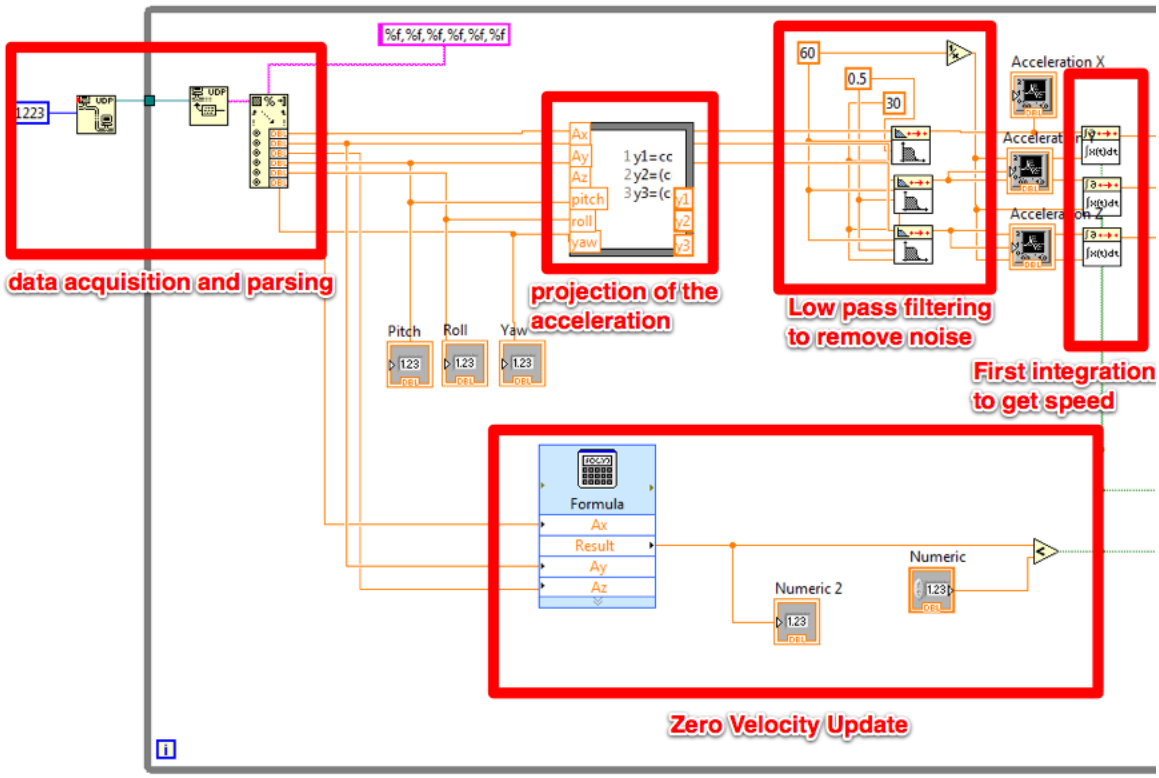

Figure 4.7. First part of our Virtual Instrument.

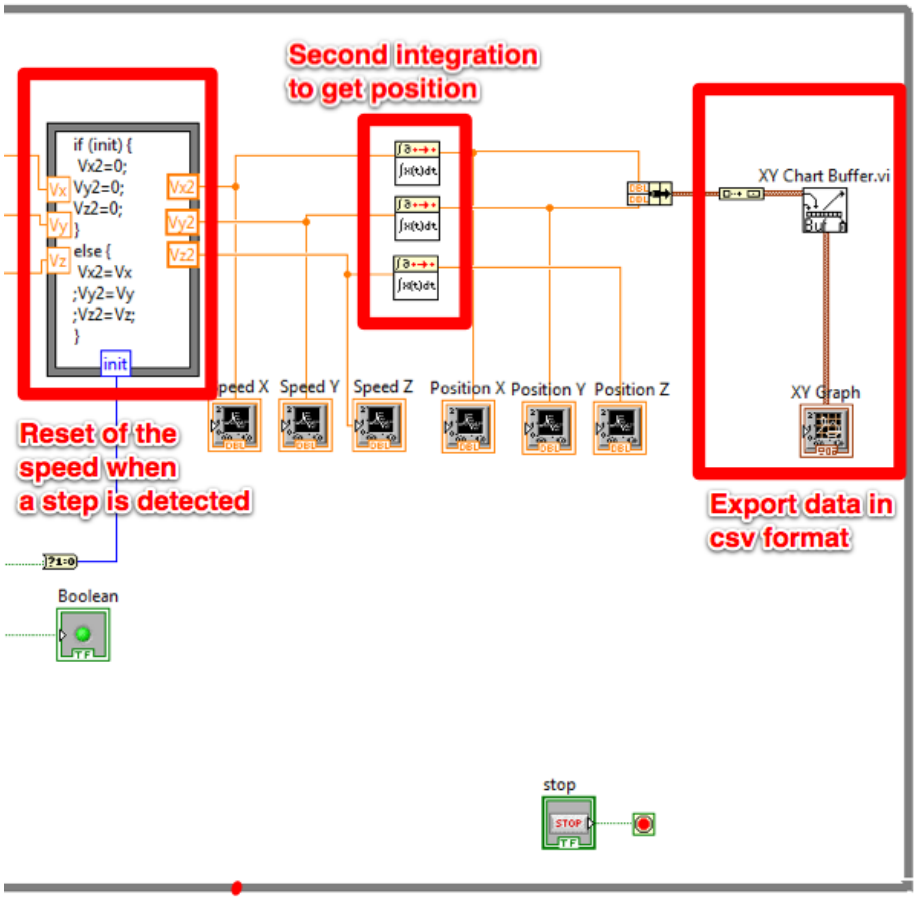

Figure 4.8. Second part of our virtual instrument

4.3.4 Initial calibration

A step of calibration is necessary for the following operations:

- Compute the angles of rotation from the rotation rate provided by the gyroscope we need to define an origin for those angles.
- We want the pitch and roll of the iPhone to be zero when the iPhone is parallel to the ground with its face up. To do so we need to calibrate the pitch and roll accordingly. Using the accelerometer and a high pass filter we can get the

direction of the gravity expressed in the referential of the iPhone. Then, using basic trigonometric functions we can get the corresponding pitch and roll of the iPhone shown in the equations below.

- Get the initial speed. The constant appears in equation (4.1) and corresponds to an integration constant.

- Get the initial position. The constant appears in equation (4.2) and corresponds to an integration constant. To make the results useful, we will need a known initial starting position. This portion will be accomplished during the path planning and navigation portion.

4.4 Results and Conclusions

The next section includes our experimental results and conclusions regarding the results.

4.4.1 Preliminary results

Our prototype was used to obtain results using only the sensors from the fourth generation iPhone: 3-axis accelerometer and 3-axis gyroscope. The measures taken during several of our test runs are shown in Fig: 4.9 - 4.10.

For the initial run, we moved in a straight line with the iPhone strapped tightly to our 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: 4.9 and Fig: 4.10. We used a simple low pass filter 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.

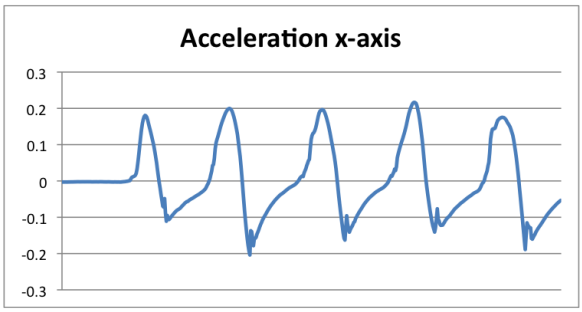

Figure 4.9. This figure shows the accelerations along the x axis (g)

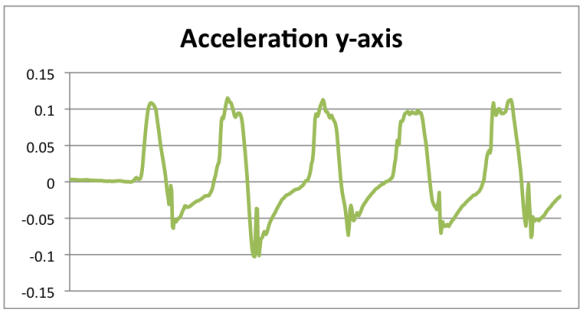

Figure 4.10. This figure shows the acceleration along the y axis (g)

On the two figures above 4.9 4.10 we can clearly observe the periodicity of the acceleration when the person is walking.

4.4.2 Final experiment & Evaluation

4.4.2.1. System with the iPhone forwarding data to LabVIEW

The setup for the final experiment was the following:

- 5 subjects walked on a 4 meters straight line with the phone strapped to their foot. The iPhone and the macbook running Labview were connected to the same network.
- The macbook was computing in real time the position of the user.
- We sampled the sensors at a frequency of 60hz (We determined this frequency empirically by observing the steadiness of the acceleration signal).
- The data is being filter using a Butterworth low pass filter (Zhongshen, Low, & Filter, 2007) with a cutting frequency of 5hz, the movement we are trying to capture (movement of a foot) is low frequency that is why we chose a cutting frequency that low.
- The Zero Velocity Update is implemented.
- The positioning data were exported in .csv and graphed with excel.

The results were definitely not accurate due to the poor quality of the sensors embedded in the iPhone and we selected three runs that are representative of the problems we encountered. The following figure 4.11 present the measured position in 2D for those three runs.

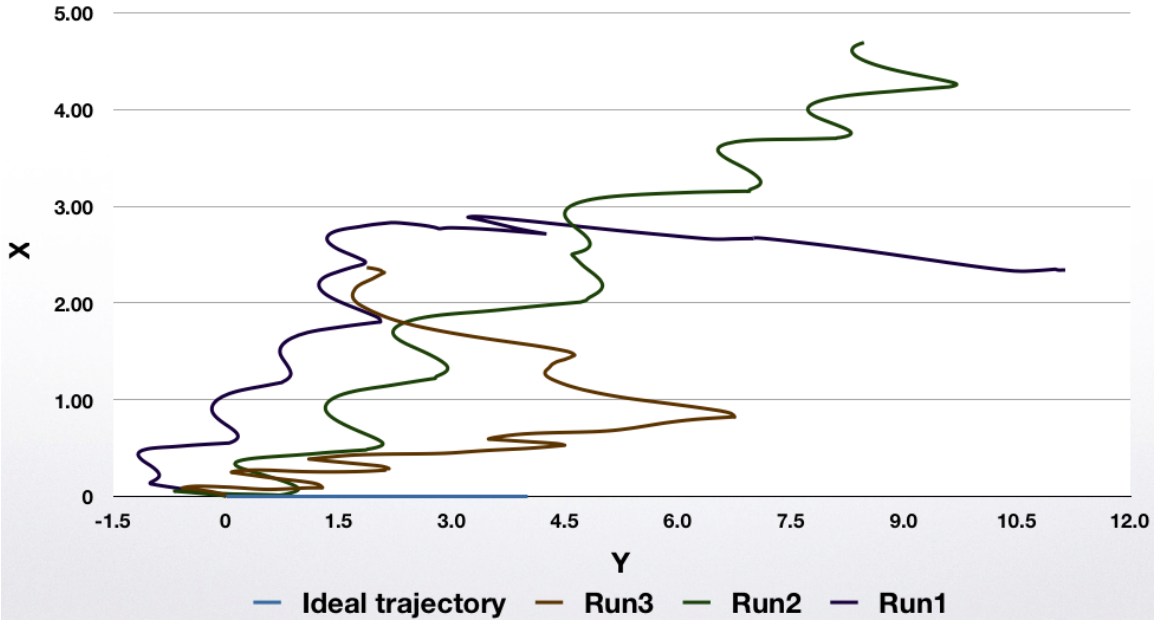

Figure 4.11. This figure shows the acceleration along the x-axis

These results shows different issues we have with the sensors embedded in the iPhone:

- Our experiments yielded longer distances than the accurate 4meters, resulting in an error up to 200%. We can explain this problem by the fact that the accelerometers on the iPhone 4 are very bad at detecting small acceleration values and most of the time goes from a null reading to a big value of acceleration. The detections is much more accurate for bigger accelerations but unfortunately the motion of the foot doesn't generate big accelerations except the peak when you put your foot on the ground.

- The X and Y axes of the accelerometers on the iPhone are interfering with each other. In this experiment we should have measured acceleration only along the Y axis as our test user was walking along it, however we can see that we also have acceleration on the X axis for the three runs. An acceleration even along

one axis of the phone always generate parasite acceleration on the other axis because of the quality of the sensors. We could have filtered it out simply by saying that the phone is always moving in the direction of the walking, but in that case we wouldn't be able to detect if the pedestrian does a left or right step.

- The accelerometers always measure a negative acceleration before measuring a positive one. In this experiment we should have observed only positive accelerations as our test users were moving forward regarding the Y axis, however we always observed a peak of negative acceleration right before getting our positive acceleration. And because we have a low pass filter this tendency to measure a backward movement takes time to dissipate and in some extreme case like the run 3 we even measured a backward movement. This initial peak might seem to be easy to filter but actually it is very inconsistent and difficult to detect automatically.

4.4.3 Conclusion

The results we obtained in this research were disappointing because of the poor quality of the sensors on the iPhone 4. As we saw a system with a double integration is so sensitive to noise that we would definitely need more accurate sensors to obtain good results. However, we developed a complete architecture to experiment on the sensors of smartphones for indoor positioning and maybe that in a few years the sensor quality on the phones will be good enough for our system to work precisely. I must also add that this research was warmly welcomed when presented at the Sensors Application Symposium 2011, as most of the focus right now is on pedometers, people were really interested to ear about our approach using inertial navigation and the conclusions we made from this experiment.

4.5 Future Work

The main future work for this project is to keep experiencing on it with newer generation of smartphones and try to refine the quality of the filtering. Also, it would be interesting to use a Kalman Filter (Balakrishnan, 1978) to combine this method with other positioning sources as GPS, Wi-Fi, RFID.

5. GRID METHOD USING RFID TAGS FOR POSITIONNING

5.1 Introduction

For this part of the study we approach the indoor positioning problem from another prospective. As we saw previously, it is difficult to create an inertial navigation system that stays accurate for a long time. A way to solve this problem would be to have landmarks that could be used to reset periodically the drift in the measure.
In this part we are going to study a landmark based positioning methods using RFID tags, I mentioned the relations it could have with indoor positioning, however we are going to study the landmark approach independently from the inertial navigation approach. This project will be focused on achieving the positioning of a person in an indoor environment using the information provided by the RFID reader of a smartphone.

5.2 Principle of the method

5.2.1 RFID tags positioning data

The principle of this method is very simple. RFID tags are placed regularly in space and each of them contains information about their positioning, this information can be:

- Latitude, longitude and optionally altitude
- x, y, z coordinated regarding a certain reference of the room

- Or other information such as floor, room number ...

The choice of the data you put in the RFID tag depends on how you want to use the system. For instance if you want to know if a room is occupied, you only need to input the room number in the tag as if the tag is scanned you have the information you want. However, if you want to be more precise and identify the position of a person in the room you will have to use multiple RFID tags per room and input more precise positioning data such as x, y, z coordinate or latitude, longitude, altitude.
In this study we decided to input the latitude and longitude into the tag as we want to be able to position a user on the map based on the reading provided by the RFID tag.

5.2.2 RFID tags as landmarks

Figure 5.1 presents how the data inside the RFID tag is used to position the user.

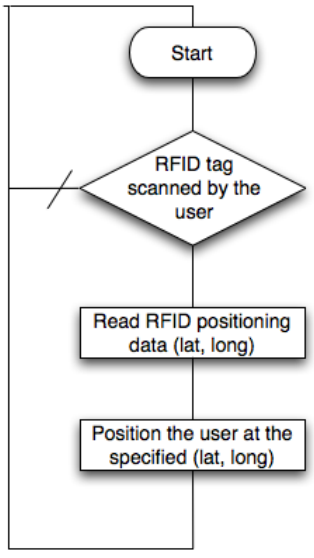

Figure 5.1. Positioning process using RFID landmarks

Every time the user scans and RFID tag, the content of this RFID tag is read and the user is positioned at the latitude and longitude indicated in the tag.

5.3 Implementation of the method

5.3.1 Requirements

We implemented the positioning method presented above with several operational constraints in mind, the tool developed must be:

- Executed on a smartphone designed for the general public.
- Easy to use.
- Self contained, it must not require external infrastructure to process the data and use exclusively embedded sensors on the phone.
- Developed on a platform that allows an easy development of applications around this positioning system.

5.3.2 Hardware used

5.3.2.1. Phone

There are not many general public smartphones having an embedded RFID reader. For this study we used a Nexus-S 5.2, a smartphone running Android and designed both by Samsung and Google.

Figure 5.2. Nexus-S smartphone

The Nexus-S corresponds to all the constraints presented above in section 5.3 as it is general public, embed an RFID reader and runs on the Android platform which has a very extended Software Development Kit.

5.3.2.2. RFID reader

The Nexus-S contains an embedded sensor able to both read and write on RFID tags. The sensor embedded is the NFC NP65 (see figure 5.3) manufactured by the company NXP semiconductors.

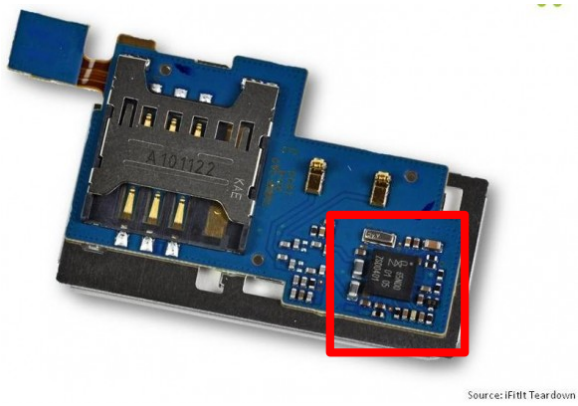

Figure 5.3. NFC NP65 embedded in the Nexus-S

This component doesn't implement the full reading and writing capabilities of the RFID standards. The NFC NP65 has been designed, as it's name indicates it, to implement only the NFC subset of the RFID standard.

NFC (Steinmeier, 2008) stands for Near Field Contact and describes the short range interactions between an RFID tag and a reader/writer. Typically, an NFC RFID tag can't be read if the reader is more than 4 inches (10 centimeters) away from the tag. This is both a technical limitation so that the reader does not require too much energy which is ideal for smartphones but it is also an operational requirement. Indeed a typical use of NFC is online payment therefore to reduce the risks of having the communication between the reader an the tag intercepted, the range has been reduced.

This limitation is both and advantage and an inconvenient for this project. As we determine the position of the user regarding the position of the tag, we want the user to be close of the tag when it is detected so that we get a good accuracy. However, 10 centimeters is a very small distance and in the case of NFC tags embedded in the floor of a room, if the user keeps its phone in hand or in its pocket, the tag won't be detected. An ideal solution would be to have tags that emit a very directional signal and with a longer range but these kind of tags doesn't exit yet.

In this project we also use the NFC Data Exchange Format (NDEF) specification (Specification, 2006), a common data format which supports four record types: text, URI, smart poster and generic control. Our positioning data will be stored as text.

5.3.2.3. RFID tags

The tags are basic passive NFC compliant tags, as a reminder passive means that the tags doesn't have their own source of energy to emit an RF signal, the reader will emit a signal which will induce a current in the tag and trigger the emission of the response, the reader is the energy source for the tag. These tags work in the 13.56MHz High Frequency (HF) range.

For this study we used passive RFID tags from UPM raflatac (see figure 5.4), these tags are small, can be stuck on many surfaces and cost between 0.60$ and 1.50$ depending on the quantity purchased.

Figure 5.4. UPM raflatac passive RFID tags

5.3.3 Software development

We developed a software implementing this positioning method and using the sensors embedded in the Nexus-S. The goal of this software is to present the user with mapping informations adapted to an indoor environment and then position the user using the method presented above. To develop this software we used the Java language and the Android Software Development Kit.

The context for this software is the campus of Purdue University and the indoor environments we consider are the inside of buildings around campus.

5.3.3.1. Overlaying indoor maps on a google map view

Data format : The GIS department of Purdue have floor plan maps available and they were able to give us jpeg images of these floorplans. We got the floorplan of the Lawson and Knoy building.

Figure 5.5. Knoy floor plan

Figure 5.6. Lawson floor plan

Georeferencing of the floor plan images: The second step was to transform these images into geographical data. We adopted a classical approach which consists in getting the geographical coordinates of at least two corners of the images (in our case we chose the top left corner and bottom right corner) to then be able to position it on

a map, also the images are oriented towards the north so we don't have any rotation operations to do in the software. The figure 5.7 presents the available data after the georeferencing step for the image.

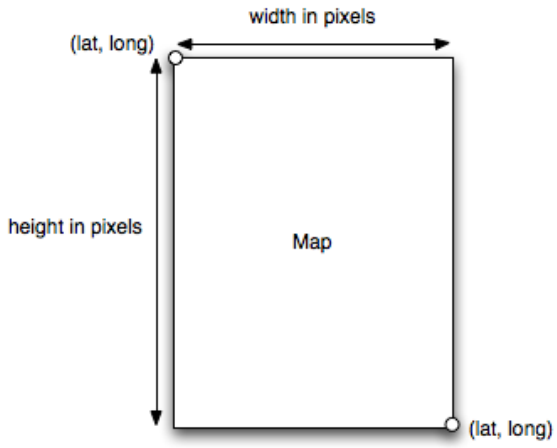

Figure 5.7. Data available for an image after georeferencing them

Displaying the image on a map After the georeferencing step, we have the data to position the image over a map view on the screen and make the position and size of the image correspond to the position and size of the building on the map. Figure **??** gives an overview of the different steps involved in the overlaying process.

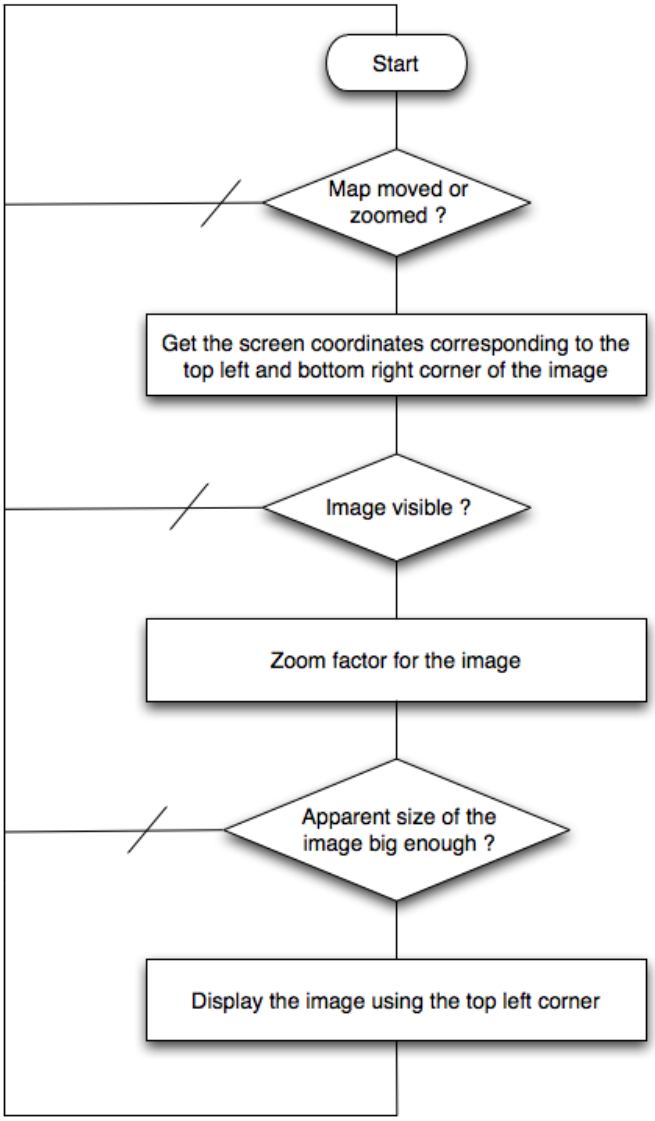

Figure 5.8. Overlaying process

- Every time the map is moved or zoomed by the user using panning or pinching gestures, the position and size of the building floor plan is computed so that it is always overlaid at the right position.
- First we get the position on the screen (in pixels) of two points with coordinates corresponding to the top left corner and bottom right corner of the image. The

Google Map API in the Android SDK has built in functions to do so and depending on the current visible part of the map on the screen will return us the screen coordinates for a certain latitude and longitude.

- Given the screen coordinates returned we can determine if the portion of the map displayed actually contains the building we want to overlay the floor plan on. For instance, if the screen coordinates for the bottom left corner are both negatives, it means that the building is currently not displayed on the map, therefore we don't do anything and wait for the next change on the map. If the floor plan can be displayed we go to the next step.

- The floor plan must be displayed at the right position but it also must have the right size. For this step we determine the ratio between the image size (in pixels) and it's actual size on the screen. To do so, we compute the size of the image diagonal as it should appear on the map (the closer you are from the building the bigger it is), and the pixels diagonal size of the image which is a fixed value.

  Following are the simple equations to compute this zoom ratio.

  First for two points point1(x_1, y_1) and point2 (x_2, y_2), we define a distance function:

$$distance(point1, point2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

  .

  Then the zoom ratio is easily defined as:

$$zoom = \frac{distance(topLeftCorner, bottomRightCorner)}{\sqrt{imageWidth^2 + imageHeigh^2}}$$

  The following figure illustrate two situations, one when the zoom ratio is superior to one because the size of the image on the screen is bigger than the size of the image and one when the zoom ratio is inferior to one because the size of the image on the screen is smaller than the size of the image.

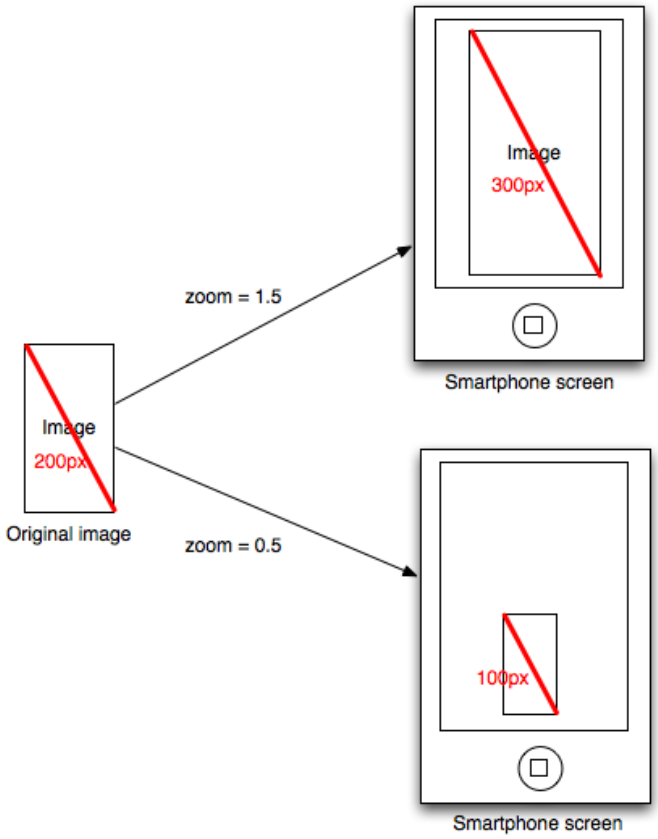

Figure 5.9. Illustration of two different zoom ratios

- Depending on the zoom ratio we then decide to draw or not the image. Indeed, for a very small zoom ratio (it happens when the zoom level of the map is very low as you see a large part of the world), the image won't be visible or won't feature enough details, so it is not worth displaying it. We set this threshold in our application to an arbitrary value of 0.1, if the zoom ratio we compute is inferior to this threshold we don't display the image.

- Finally with both the position and zoom ratio information we can draw the image at the right position and at the right size on the map.

The two following figures presents screenshots of the application where the floor plan of the Lawson and Knoy building are overlaid using the phone on a Google Map View in Aerial photos mode.

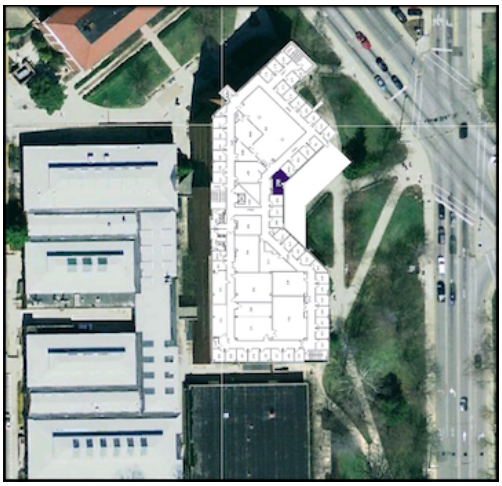

Figure 5.10. Knoy floor plan

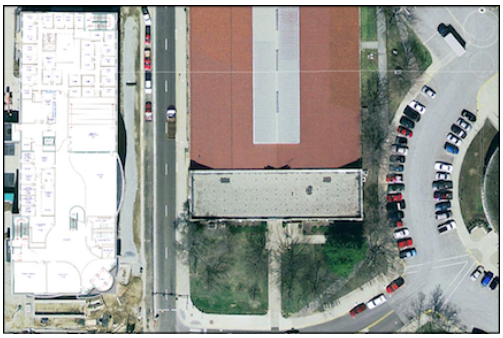

Figure 5.11. Lawson floor plan

5.3.3.2. Reading from RFID tags and displaying the corresponding position on the map

This step is necessary to retrieve the positioning data contained in the tag.

Every app in Android has the same lifecycle, after being initialized, the app wait for an event (touch on the screen, GPS update, button pressed ...) and then execute the associated activity. It then, waits for another event. If one or multiple events are received while an activity is executed they are added to the event queue and treated one at the time when the previous activity is done. The following figure present a simple state diagram of the lifecycle of an android app.

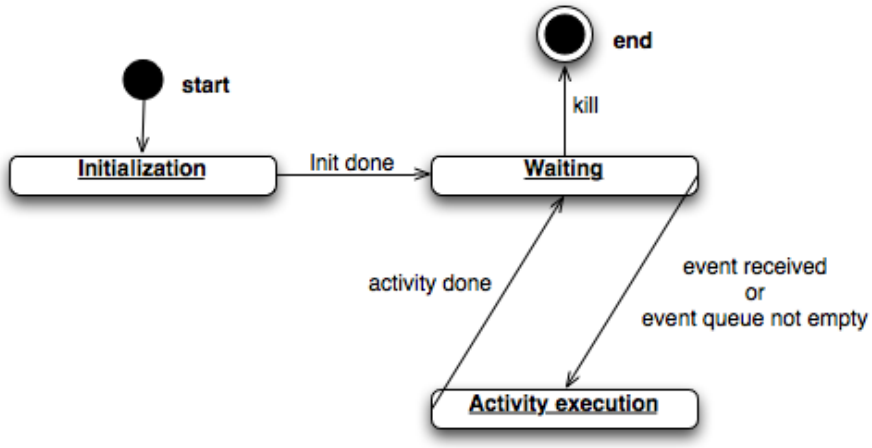

Figure 5.12. Two different

The NFC library in the Android SDK allows us to generate an event every time the RFID reader detects a tag. We then simply read and parse the content of the tag to retrieve a value for the latitude and longitude of the tag.
The latitude and longitude information give us a point that we can display on the map, we also store this point in an array to be able to generate the path the user is following.

5.4 Experiment & Evaluation

5.4.0.3. Setup of the experiment

For this experiment we want to track the position of the user in some of the corridors in the Knoy building.

RFID tags are dispatched regularly along the wall and also placed at strategic points such as intersections and doors.

5.4.0.4. Positioning data of the RFID tags

The preciseness to which we input the positioning data in RFID tag is primordial for this setup. Indeed, as stated before, the RFID tag is a landmark and when the user scans a tag its position is assimilated to the position of the tag. Therefore, if the positioning of the tag is not precise, then the positioning of the user won't be.

As our main concern is to present the user with its position on the map we focused more on having the position of the tag show up at the right place on the map than estimating precisely its coordinates. We used the following approach:

- In an indoor environment we cannot use GPS to measure the geographical coordinates of a point. We do a first measurement of the tag geographical coordinates using a paper version of the floor plan, as we have the geographical coordinates of two of the corners we can with a ruler and a few crossproducts estimate the latitude and longitude of any point on the floor plan.

- We then refine this position by scanning the marker and making sure it is shown at the right position on the map, if not we make small adjustments until it does. Doing this step we don't consider if in the end the coordinates of the tag really correspond to the coordinates of the point it is placed on, however we ensure that what the user will see on the map when it scans the tags is the position of the tag in the corridor. This way, even if the maps is positioned slightly off it's real position, the position shown on the map will be off as well regarding the global google map but won't be regarding the floor plan overlay and this is what we want.

5.4.0.5. Results

Figure 5.13 present a run of the application with the path of the user reconstructed from the different RFID tags he scanned. Each point on the path represents a tag scanned. As indicated on the figure 5.13 the placement of the tags is not random, we will explain in detail how we chose the placement of the tags.

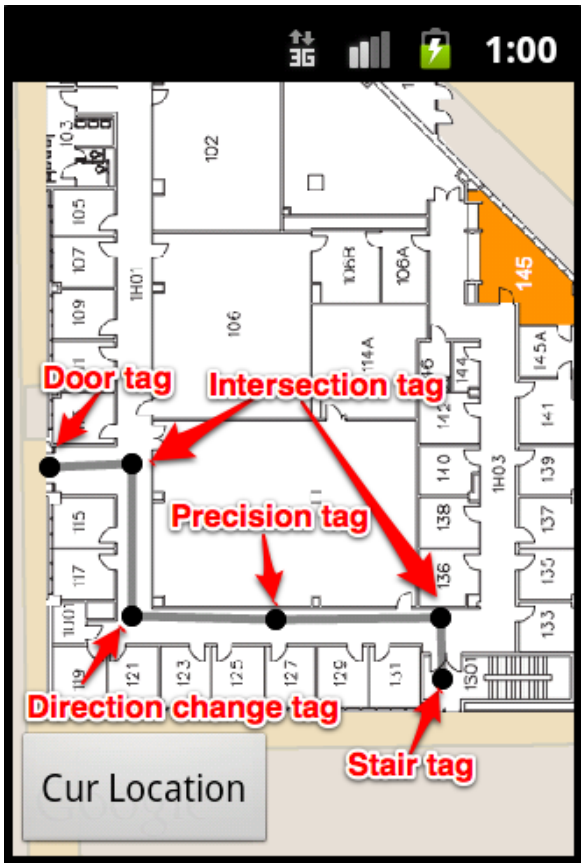

Figure 5.13. Test run in the first floor of Knoy

We disposed the tags in a way that would be beneficial for users who couldn't assess the environment around them (disabled persons, robots ...):

- Door Tags and stair tags. These tags are placed next to entry points and exit points of the floor, this is important to have them because they usually are the first tag or last tag scanned by the user in a specific environment. In our case, if

we consider the first floor of Knoy as the environment, the door tag materializes the entry point and the stairs, the exit point.

- Intersection tags. These tags are placed every time there is a possible change of direction. These tags could be used to provide the user with useful navigation informations.

- Direction change tags. These tags are placed when a change of direction is required from the user. In the figure above, the user can be warned with this tag that he has to make a turn not to go straight into the wall.

- Precision tags. When there is no change of direction possible or needed for a long distance (typically when you walk in a long straight corridor), we regularly put tags along the line to ensure the user is still following a straight line.

5.4.0.6. Evaluation

We created a system that successfully implements the positioning method presented and meet all the requirements presented previously in the subsection 5.3.1.

- It is executed on a Nexus-S smartphone which is a general public smartphone.

- It is easy to use, the user only have to launch the application and start scanning tags.

- It does not require any external infrastructure to process the data and use exclusively the sensors embedded in the phone.

- It is developed on the Android platform for which the SDK provided by Google makes it very accessible for any developer to create applications.

5.5 Conclusion & Future Work

We presented in this chapter an indoor positioning method using RFID tags as landmarks and developed an application on an Android phone that implements this method. This application is open source and it is easy to build up from it.

A part of the future work, which would be to integrate other positioning sources in this software, is actually a work in progress. This method and software are part of a bigger project dedicated to provide seamless navigation capabilities wether you are in an indoor or outdoor environment. This system integrates GPS, Wi-Fi and RFID positioning and selects the more accurate positioning source depending on the user environment.

Future work for this project could also be to include spatial constraints in the software. For now we make sure that the position displayed on the map corresponds to the position of the tag by actually scanning the tag and looking if it is displayed correctly on the map. Instead of having this manual process we could take into account spatial constraints. For instance, if our tag is not positioned precisely and make it so that the position of the user is outside of the corridor, taking into account the fact that the user should be in the corridor and not embedded in the wall could help us position the user more precisely. It is the same principle with GPS in the cars where, even if the GPS gives you a position that is off the road the display on the screen will still position you on the closest road.

6. MULTI AGENT FRAMEWORK FOR NAVIGATION SOFTWARE

6.1 Introduction & Definitions

This chapter is extracted from a paper currently under review for the 2012 edition of International Conference on Practical Applications of Agents and Multi-Agent Systems.

The two previous chapters presented two different methods for indoor positioning which is an important aspect for a navigation application but we have not talked yet about the navigation aspect of the problem. In this chapter we present a multi-agent system designed for navigation application and an implementation of this model on an iPhone. This iteration of the project focus on the multi-agent system composed of three specialized agents:

- A Path Planner in charge of the path planning tasks.
- A Tracker in charge of positioning and itinerary tracking.
- A Translator in charge of making the interface between the system and the user.

The word user represents a person or system using the multi agent system.

6.2 Design of the multi agent model

6.2.1 Overview

The system is composed of three specialized agents and a blackboard used as a repository for shared information. Figure 6.1 presents an overview of the system, which agent uses and updates the data in the blackboard and which messages they are susceptible to send.

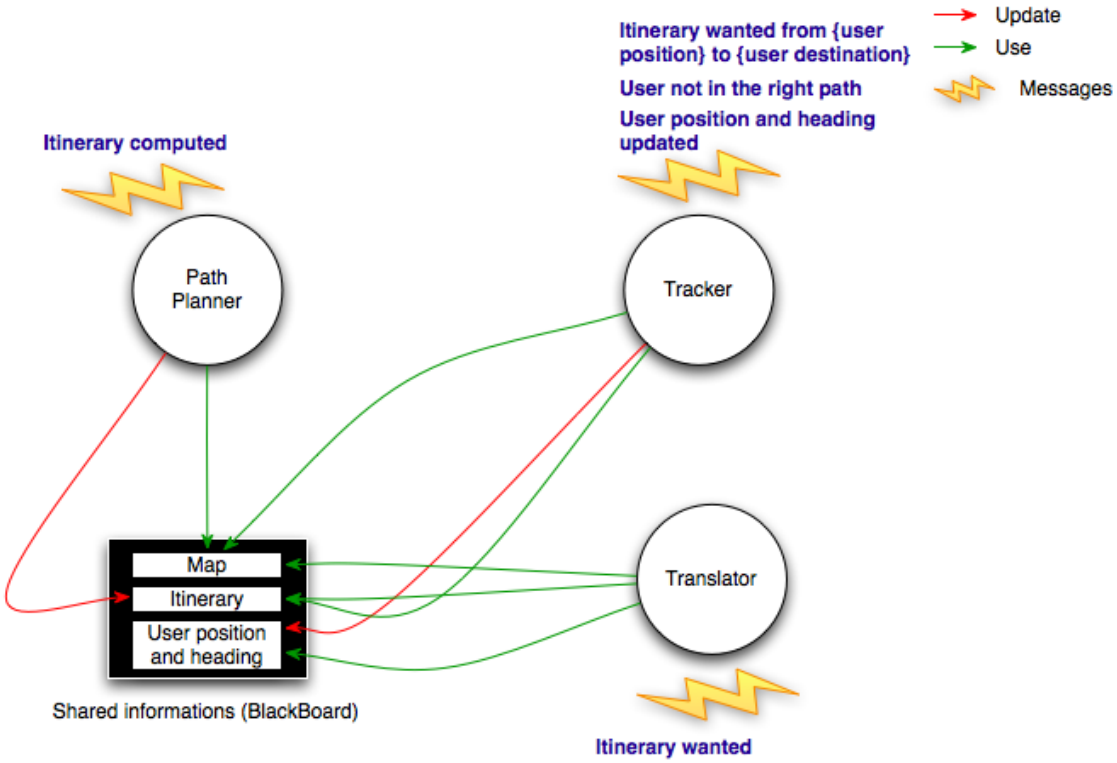

Figure 6.1. Overview of the multi-agents model

Path Planner The Path Planner is the agent in charge of computing itineraries.

Translator The Translator is the agent in charge of doing the interface between the multi-agent system and the user. In typical car navigation applications this agent will be in charge of displaying the mapping and navigation informations in a comprehensive way for the user.

Tracker The Tracker is the agent in charge of getting the position and heading of the user. This agent is in charge of harvesting the data from the sensors used for positioning like a GPS, or in the case of our project the gyro and accelerometer. The position information provided by the Tracker can be an absolute position but also a position corrected according to the mapping informations, for instance in the context of car navigation if the absolute position is outside a road we might want to correct it so that the user is placed on the closest road. Another function of this agent is to track if the user is following the itinerary, if the user deviates from the computed path the agent will issue the right messages to trigger the recomputing of the itinerary.

Shared informations (Blackboard) The blackboard is a storage space where agents can put information shared by all the agents in the system. Each agent is able to read and update information. The information stored are:

- The map which is used by the Path Planner to compute the itinerary, by the Tracker to correct the position if needed and by the Translator in case it is needed to provide mapping information to the user.
- The itinerary, this data is created and updated by the Path Planner when it computes the itinerary. It is used by the Tracker so that it can make sure that the user is on the right itinerary and by the Translator in case it is needed to provide informations about the itinerary (direction of the next turn, distance remaining ...).
- The user position and heading, this data is updated by the Tracker and is used by the Translator to provide the user with information regarding its position.

6.2.2 State diagrams

This section present the state diagrams of each agents.

- rcv:“A message” + (data1/description1 data2/description2 ...) correspond to the reception of the message “A message” along with some data.
- snd:“A message” + (data1/description1 data2/description2 ...) correspond to the broadcasting of the message “A message” along with some data.

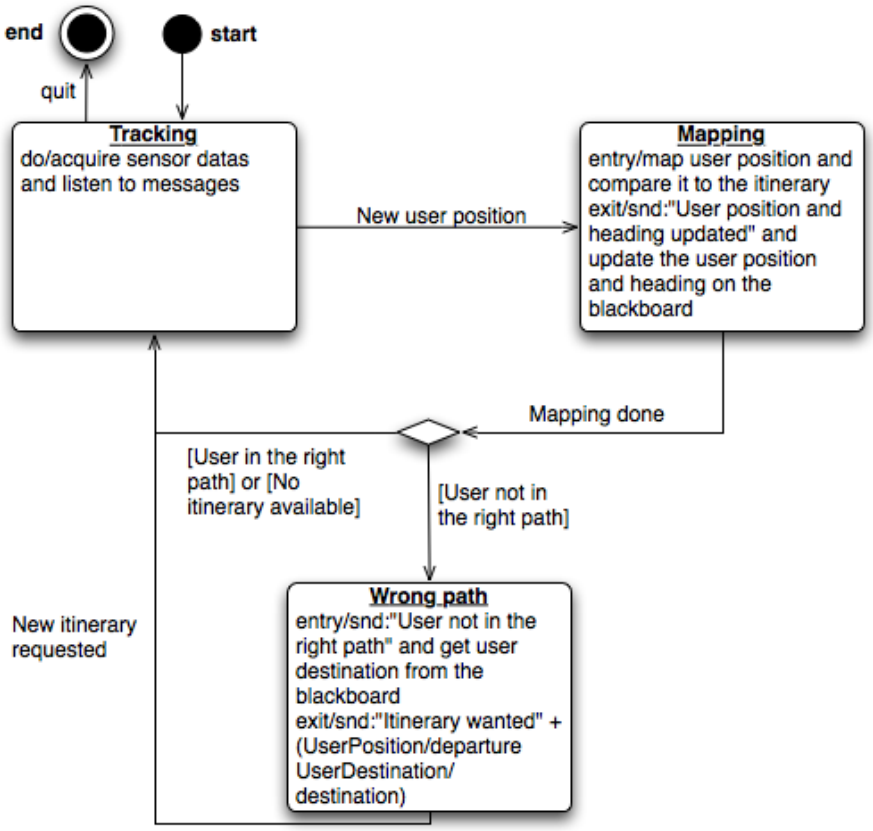

Figure 6.2. State diagram of the Tracker

Tracker The Tracker (see figure 6.2) can be considered as the clock of the multi-agent system, indeed our system evolves when a new position for the user is acquired,

and the Tracker is the agent in charge of detecting the position of the user.
In the regular flow, the Tracker goes back and forth between two states: **Tracking** and **Mapping**. In the **Tracking** state the agent acquires the sensor data and computes the position of the user (maybe after steps of filtering) in the appropriate coordinate system. When a new user position is computed the Tracker goes to the **Mapping** state where the position is adjusted according to the mapping informations and the context of the navigation. When the Tracker enters the **Mapping** state, it also compares the position of the user to the itinerary. When the mapping of the user position is done, the Tracker sends a message "User position and heading updated", and if the user is in the path corresponding to the itinerary or if no itinerary is available, the Tracker goes back to the **Tracking** state. If the user is not in the right path the Tracker goes to a **Wrong Path** state, when the Tracker enter this state a "User not in the right path" message is sent and the Tracker gets the user destination from the blackboard. Then the Tracker requests a new itinerary by sending a message "Itinerary wanted" along with the User position as departure and the User destination, then it goes back to the **Tracking** state

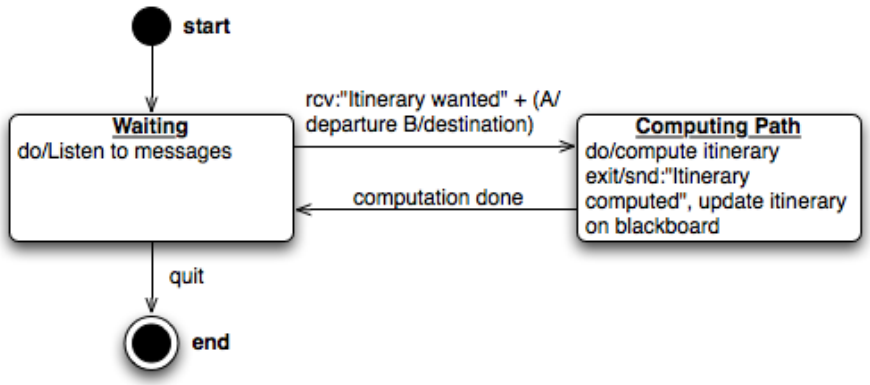

Figure 6.3. State diagram of the Path Planner

Path Planner The Path Planner (see figure 6.3) has two states: **Waiting** and **Computing Path**. In the waiting state the Path Planner listen to the messages from

other agents. When a message "Itinerary wanted" and the corresponding departure and arrival data are received, the Path Planner goes into the **Computing Path** state and compute the itinerary from A to B. When the computation is done the Path Planner goes back into the waiting state, sends an "Itinerary computed" message and update the corresponding itinerary on the blackboard.

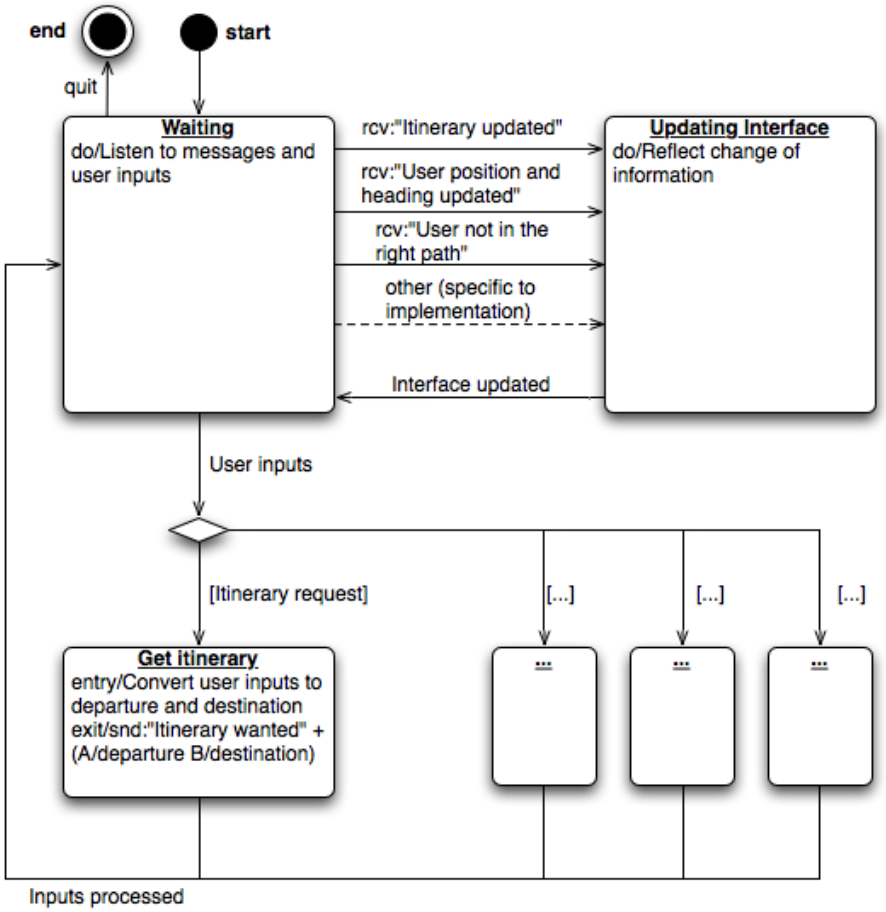

Figure 6.4. State diagram of the Translator

The Translator The Translator (see figure 6.4) starts in a **Waiting** state and listens to messages from other agents and user inputs. When one of the three messages listed is received, the Translator goes into the **Updating Interface** state where the corresponding actions are taken to reflect the change of information (visual changes,

sounds, vibrations ...). Other transitions from the **Waiting State** to the **Updating Interface** state can be implemented to fit the specific characteristics of the system implementing this model.

When a user input is detected the Translator goes into a specific state according to the user input, here we describe one state corresponding to the input of an itinerary, the **Get Itinerary** state. In the **Get Itinerary** state the Translator converts the user inputs to two positions on the map corresponding to the departure and the destination (the departure can correspond to the user position, in that case no input is required). Then when the inputs are processed a message "Itinerary wanted" along with the departure and destination selected are sent and the system goes back in the **Waiting** state. Other states can be implemented to fit the specific inputs of the system implementing the model.

6.2.3 Agents organization

Each agent have the same communication capabilities: they are able to send a textual message along with some data. The messages sent by the agents are broadcasted, all the agents have exactly the same communication capabilities and listen to all the broadcasted messages.

There are no specific coordination steps in this model, the organization is implicitly know by the agents as they share the same unambiguous communication protocol, and know which actions they have to associate to each messages.

To share data the agents can use the blackboard for datas that are central in the model and have an important lifetime. For transient data, agents can broadcast data along with the messages they send. This is used when a new itinerary is computed, the message is broadcasted along with the departure and arrival.

6.3 Implementation

We developed a simple navigation application dedicated to pedestrians using this multi-agent system. This application has been developed on an iPhone 4. We chose this platform as it is the one we are also developing an indoor positioning system for. The three agents are embedded in the iPhone and have been implemented in C and Objective-C we also used the CocoaTouch framework "Apple developer documentation" provided by Apple.

This application will serve as a foundation for our final project.

6.3.1 Data

The map and itinerary are represented as a directed graph. In this application we use a simple text representation of the list of nodes and the list of edges, along with the weight of each edges to define the graph.

Figure 6.5 presents the textual representation of a graph, these datas are usually stored in a text file.

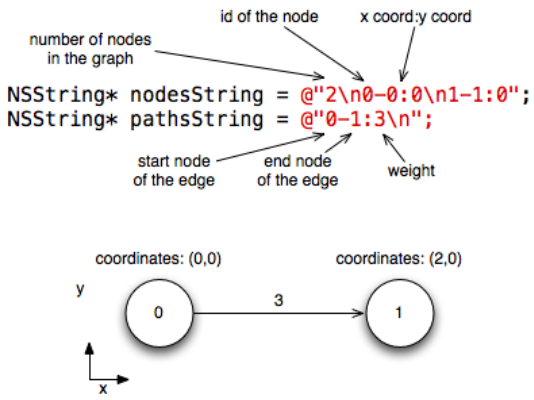

Figure 6.5. Textual representation of a graph

6.3.2 Class diagram

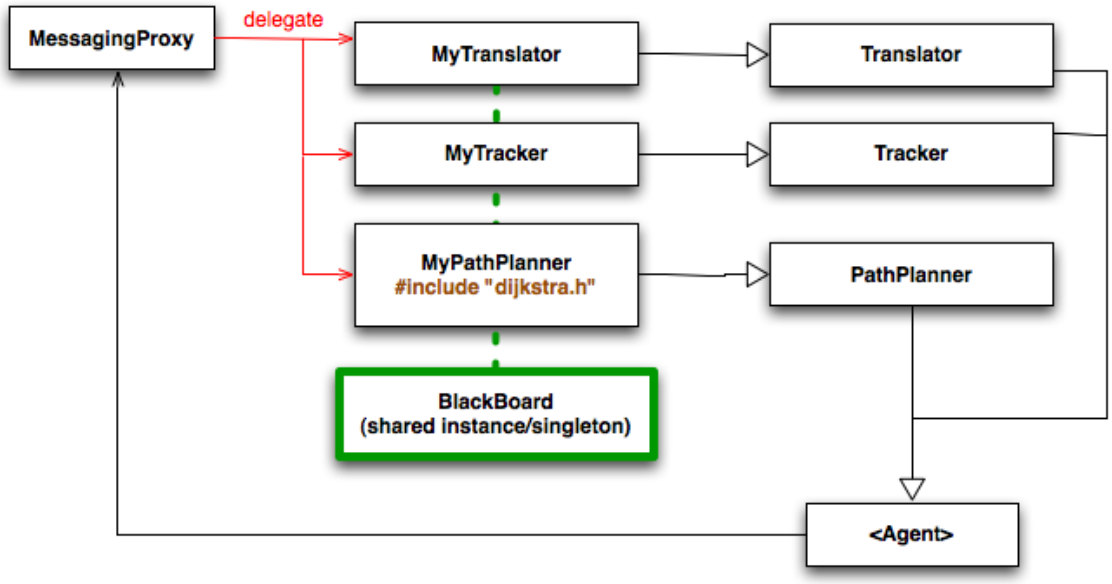

Figure 6.6. Class diagram of the system

The blackboard is a storage place shared by all the agents we use the SharedInstance/Singleton pattern to implement it, this way we make sure there is only one instance of the blackboard.

The Agent class describes the common behavior the agents of our system share. Especially to be able to communicate between each others the agents have to use an instance of the class MessagingProxy whose behavior will be described in the next subsection.

Each agent is then described by a class: Translator, Tracker, PathPlanner. The classes define which general actions the agent must implement, its different states, and the binding between the messages and the functions.

The Translator, Tracker, PathPlanner and Agent could be defined as abstract classes but this is not offered by Objective-C.

The classes MyTranslator, MyTracker, MyPathPlanner implements the actions described by their superclass.

Using this model we have a level of abstraction allowing us to modify easily the implementation of each actions depending on the context of the application.

6.3.3 Implementation of the communication layer

The MessagingProxy class can be considered as the messaging interface of each agent. This component implements the communication capabilities of our agent: Being able to broadcast a textual message and some data using one or more communication channels available on the targeted platform.

In our application we use a broadcasting mechanism part of the CocoaTouch framework called NSNotification "NSNotification class reference". NSNotification can broadcast an integer identifier and a dictionary containing any number of objects.

Figures 6.7 and 6.8 describes how initial messages and datas are "translated" into NSNotification objects.

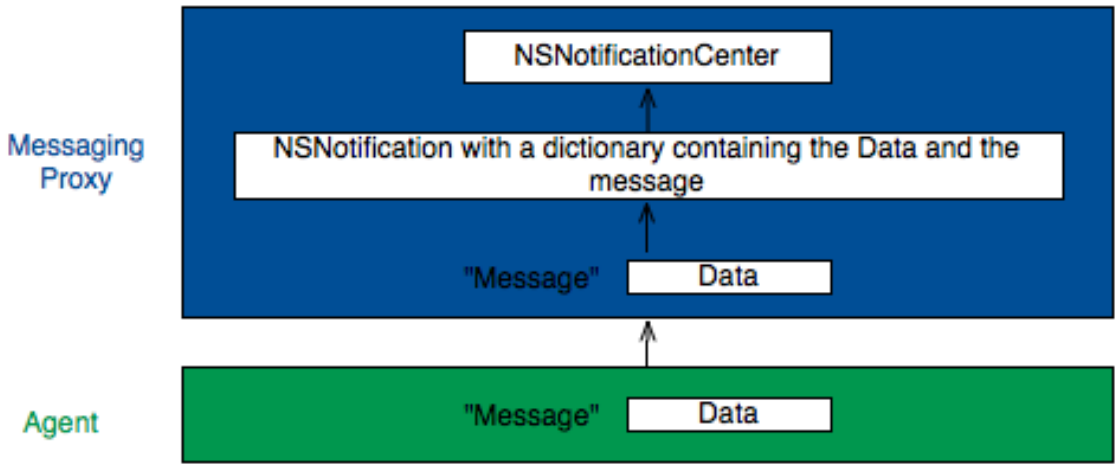

Figure 6.7. Expedition of a message

To transform the message and data the agent wants to send into an instance of NSNotification the proxy places the message and the data in a dictionary, then encapsulates this dictionary in an NSNotification object which is then dispatched by the NSNotificationCenter.

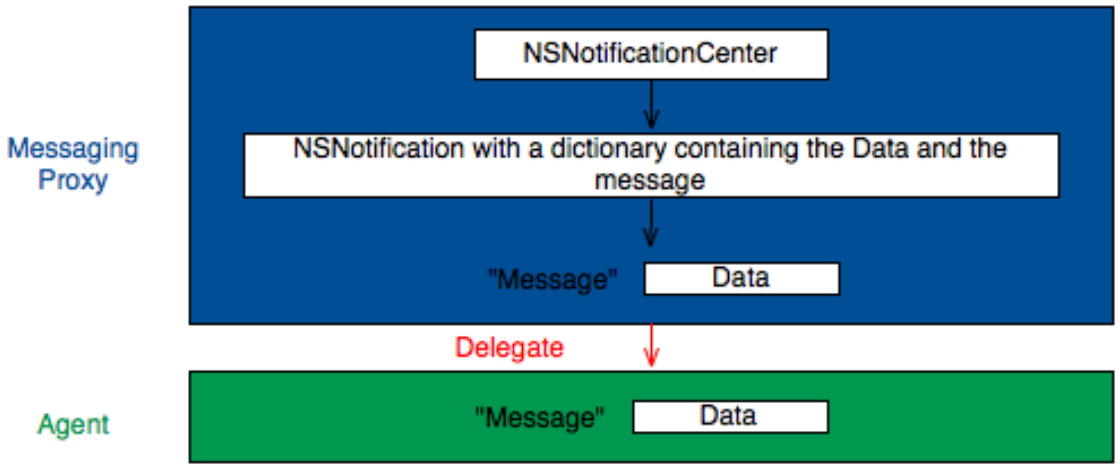

Figure 6.8. Reception of a message

When a message is received the reverse process takes place.
The messaging proxy listens to the NSNotifications broadcasted using the NSNotificationCenter, then the proxy brakes down the NSNotification object received into the message and data.
We then use the delegate pattern to allow the agents to be informed by their MessagingProxy that a new message and data has been received This messaging architecture makes it easy to implement other communications channels, to do so we only have to implement in our proxy a way to "encode" and "decode" the message and the data in a way that can easily be transmitted through this communication channel.

6.3.4 Implementation of each agent

Path Planner : The Path Planner uses the Dijkstra algorithmCormen, Leiserson, Rivest, and Stein (2001) to find the shortest path between the departure and arrival selected by the user.

Tracker : For this application the positioning data is simulated. The fake path followed by the user is defined when the application starts (this path can or cannot correspond to the itinerary) and the application virtually changes the position of the user with a certain frequency.

Translator : In this application the end user is a human, so the Translator is equivalent to a Human Computer Interface. It presents visual information on the screen about the map and the itinerary, a more precise description of the different parts of the Graphical User Interface will be done in the results subsection (6.3.5)

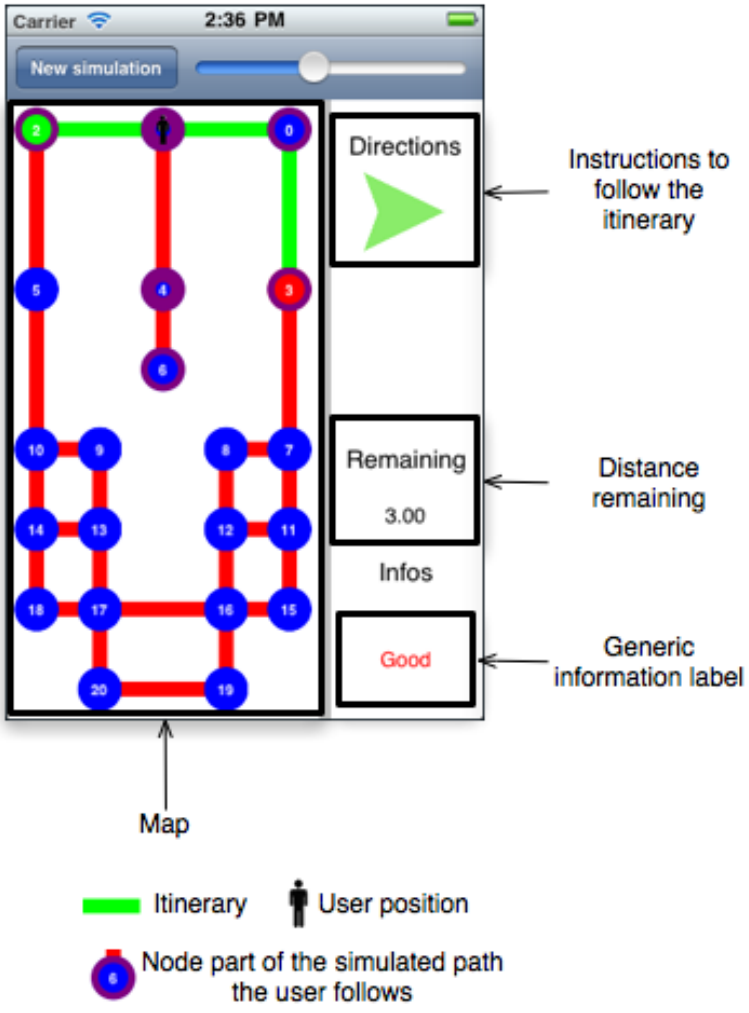

Figure 6.9. Graphical interface of our system

6.3.5 Results

This section presents screenshots of our application in five specific situations.

6.3.5.1. Initial itinerary selection

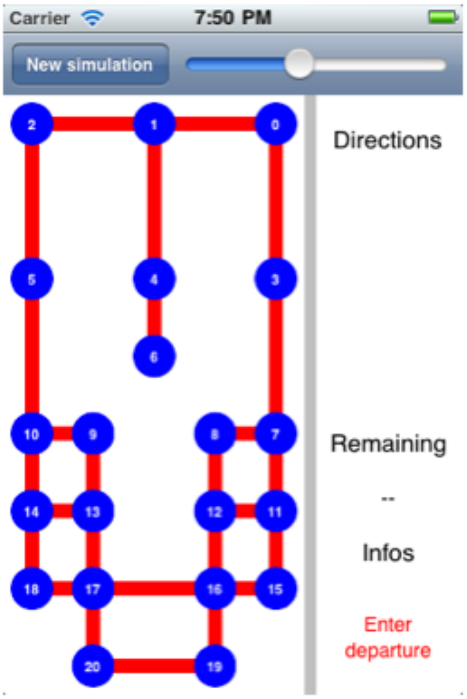

Figure 6.10. Welcome page

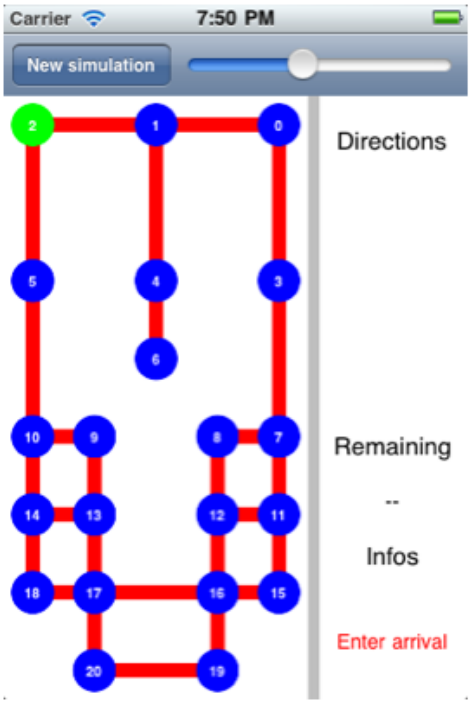

Figure 6.11. Departure entered

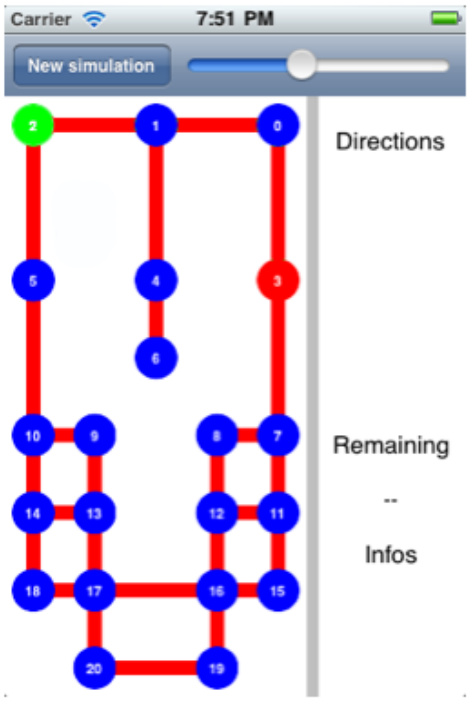

Figure 6.12. Arrival entered

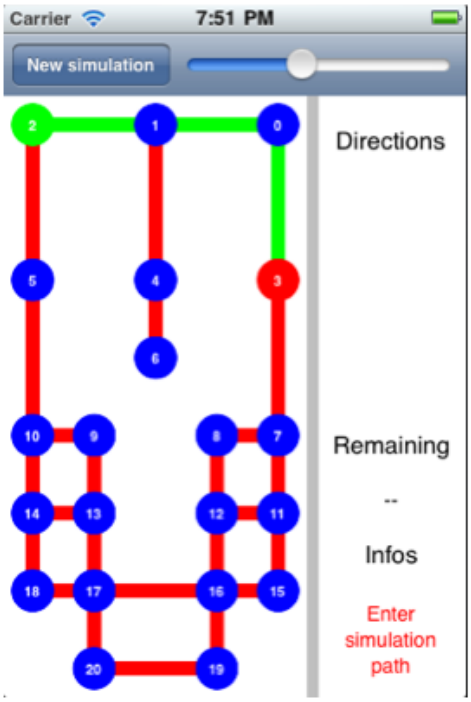

Figure 6.13. Itinerary computed

When the application starts the user is asked to enter the destination and departure (figure 6.10). The user touches the screen to enter the departure (green node figure 6.11) and arrival (red node figure 6.12) of the desired itinerary, the Translator is in the **Get itinerary** state at this moment. Then the Translator sends the message "Itinerary wanted" along with the departure and arrival selected and goes back to the waiting state.

The Path Planner receives the "Itinerary wanted" message along with the departure and arrival, goes into the **Computing Path** state and compute the shortest path between the departure and arrival. When the computation is done, the Path Planner goes back into the **Waiting** state and send the message "Itinerary computed".

The Translator receives the "Itinerary computed" message, goes into the **Updating Interface** state and displays the itinerary on the map (green path figure 6.13).

6.3.5.2. Simulation path

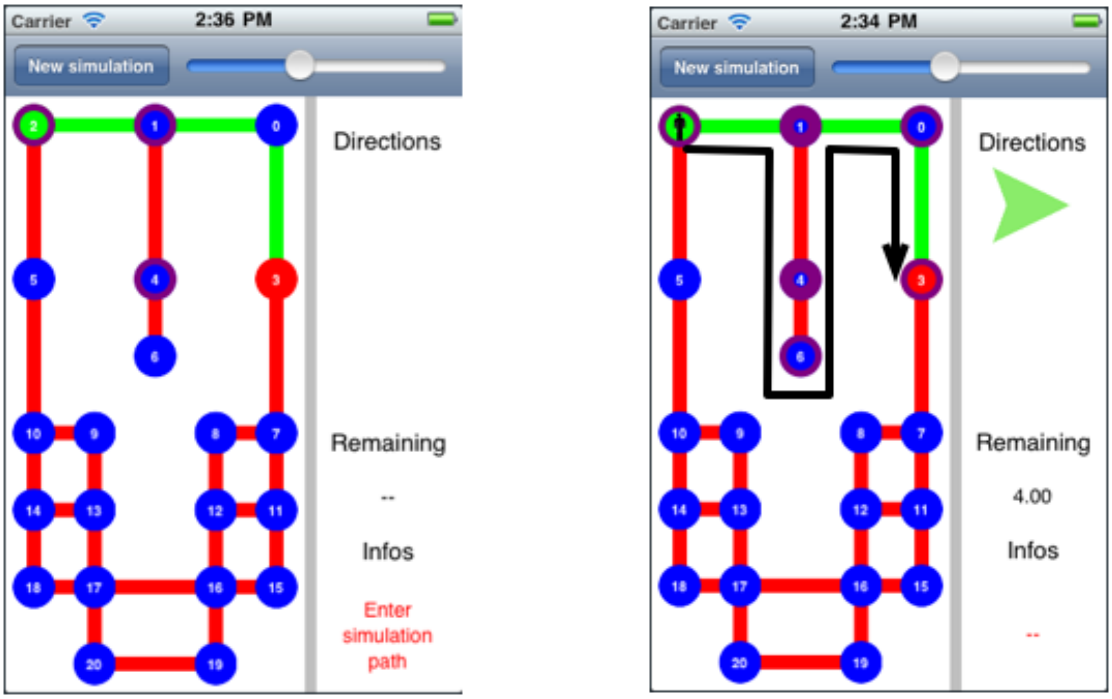

Figure 6.14. Simulation path entered

The simulation path can be configured by touching on the screen the nodes the simulated path is going to be composed of (purple nodes figure 6.14). The Translator goes into the **Get simulation path** state and send a message "Simulation path acquired" along with the data corresponding to the path. The Tracker receives this information, stores it and starts sending positioning signals corresponding to the simulated data. The states and messages corresponding to this step are not described in the model as they are artifacts of the simulation.

You can notice that the simulation path doesn't correspond to the itinerary, this way we will be able to observe how our system behave when the user steps out of the itinerary.

6.3.5.3. Normal step

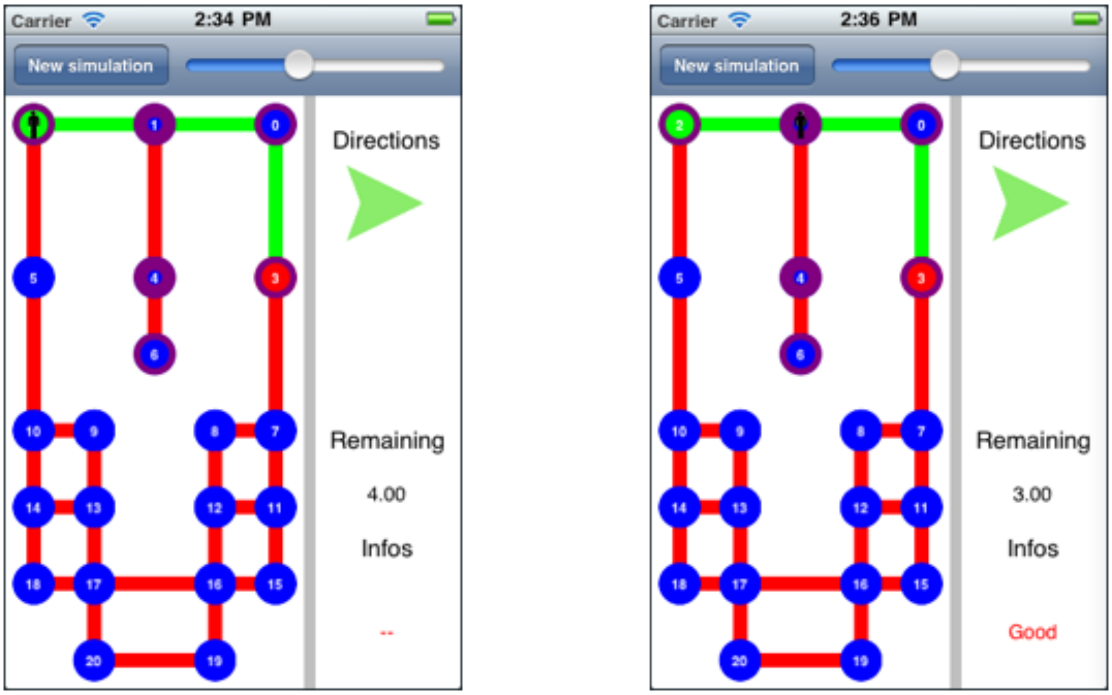

Figure 6.15. Normal step

A normal step corresponds to a change in position where the user stays on the computed itinerary.

The Tracker initially in the **Tracking** state generates a new position and heading for the user. The Tracker then goes in the **Mapping** state and compares the position of the user to the itinerary. The user being on the itinerary, the Tracker sends a "User position and heading updated" message and updates the user position and heading

on the blackboard.

The Translator receives the message and goes into the **Updating Interface** state. The new position of the user is displayed, the direction to turn to at the next intersection and the remaining distance are updated (see figure 6.15).

6.3.5.4. Step outside of the itinerary path

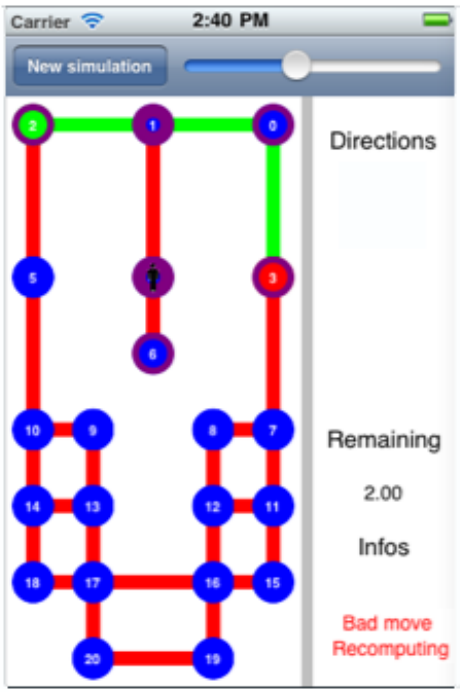

Figure 6.16. Step outside of the itinerary path

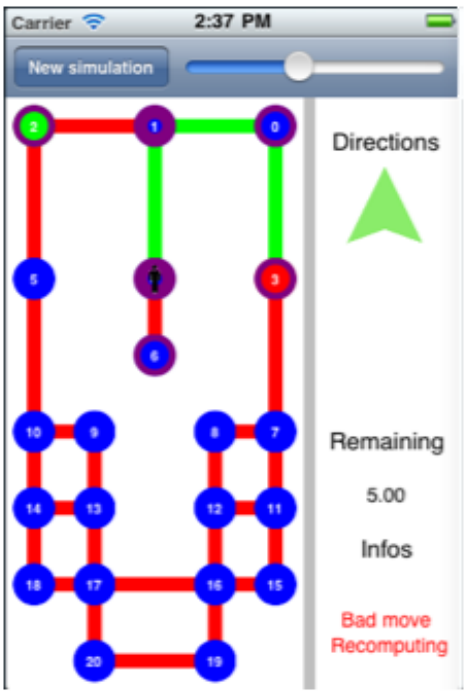

Figure 6.17. New itinerary computed

The Tracker behaves the same way as for a normal step (see 6.3.5.3) when in the **Tracking** state and then **Mapping** state, the same for the Translator when it receives the "User position an heading updated" message. However as the user is not

in the right path the Tracker goes into the **Wrong Path** state sends a "User not in the right path message", gets the destination of the itinerary from the black board and sends the message "Itinerary wanted" along with the User position as departure and the User destination, then it goes back to the tracking state.
The Translator reacts to the "User not in the right path message" by going into the **Updating Interface** state and displays an alert on the screen (red text figure 6.16). The Path Planner reacts the same way as described for the initial itinerary selection (6.3.5.1). When the Translator receives the "Itinerary computed message", it goes to the **Updating Interface** state, displays the itinerary and updates the direction to turn to at the next intersection and distance remaining (see figure 6.17).

6.4 Evaluation

In this part of the study we developed a model. It is very difficult to asses the external validity of the model because it would require to actually consider all the implementations of such a model. However, in this study we stayed general enough and didn't use any platform specificities to make sure this model could be used on multiple types of infrastructures. The internal validation of the model is achieved with the successful development of a prototype on iPhone 4.

6.5 Conclusion & Future work

We presented in this chapter a multi-agent system designed for navigation applications. We also presented an implementation of this model on a smartphone, however this implementation presents only one of the many possibilities to implement this model. Nowadays navigation applications are used in a lot of different contexts and using a multi-agent system provides a solid and highly flexible framework to develop these applications. As stated in the introduction this model has been extracted from the development of an indoor positioning system designed for the smartphones. In that regard our future work consists in moving from a system designed for a theoret-

ical environment with simple map datas and simulated positions to a system using the indoor positioning method we are developing and actual geographical datas of a building. This will also require improvements regarding the way we achieve path planning so that we are able to handle larger graphs, this can be achieved by embedding with the geographical data the predecessor list for certain precomputed paths and so doing real time computation for only small parts of the itinerary. Finally we are planning to improve the communication protocol to get a better error handling in the system.

LIST OF REFERENCES

(2010). Retrieved from http://developer.apple.com/library/ios/#documentation/CoreMotion/Reference/CoreMotion_Reference

Apple developer documentation. (n.d.). Retrieved from http://developer.apple.com/library/ios/navigation/

Balakrishnan, A. V. (1978). The kalman filter. *The Mathematical Intelligencer*, *1*(2), 597–92. Retrieved from http://www.cs.unc.edu/%7Ewelch/kalman/

Constandache, I., Choudhury, R., & Rhee, I. (2010). Towards mobile phone localization without war-driving. (pp. 1 –9). doi:10.1109/INFCOM.2010.5462058

Cormen, T., Leiserson, C., Rivest, R. L., & Stein, C. (2001). *Dijkstras algorithm*. MIT Press and McGraw-Hill.

Dekel, A., & Schiller, E. (2010). 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* (pp. 393–394). Lisbon, Portugal: ACM. doi:http://doi.acm.org/10.1145/1851600.1851681

DeLoach, S. (2006). Engineering organization-based multiagent systems. In A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet & A. Romanovsky (Eds.), *Software engineering for multi-agent systems iv* (Vol. 3914, pp. 109–125). Lecture Notes in Computer Science. Springer Berlin / Heidelberg. Retrieved from http://www.springerlink.com/content/kq11151166474040/

Deloach, S. A., Wood, M. F., & Sparkman, C. H. (2001). Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, *11*(3), 231–258. doi:10.1142/S0218194001000542

DeLoach, S. A., Oyenan, W. H., & Matson, E. T. (2008). A capabilities-based model for adaptive organizations. *Autonomous Agents and MultiAgent Systems*, *16*(1), 13–56. doi:10.1007/s10458-007-9019-4

Drane, C, Macnaughtan, M, & Scott, C. (1998). Positioning gsm telephones. *IEEE Communications Magazine*, *36*(4), 46–54, 59. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=667413

Falco, J., Idiago, M., Delgado, A., Marco, A., Asensio, A., & Cirujano, D. (2010). Indoor Navigation Multi-agent System for the Elderly and People with Disabilities. In Y. Demazeau, F. Dignum, J. Corchado, J. Bajo, R. Corchuelo, E. Corchado, ... A. Campbell (Eds.), *Trends in practical applications of agents and multiagent systems* (Vol. 71, pp. 437–442). Advances in Soft Computing. Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-12433-4_52

Fu, Q., & Retscher, G. (2009). Active rfid trilateration and location fingerprinting based on rssi for pedestrian navigation. *Journal of Navigation*, *62*(02), 323. Retrieved from http://www.journals.cambridge.org/abstract_S0373463308005195

Hsu, C.-H., & Yu, C.-H. (2009). An accelerometer based approach for indoor localization. (pp. 223 –227). doi:10.1109/UIC-ATC.2009.90

Hynes, M., Wang, H., & Kilmartin, L. (2009). Off-the-shelf mobile handset environments for deploying accelerometer based gait and activity analysis algorithms. (pp. 5187 –5190). doi:10.1109/IEMBS.2009.5333715

Joho, D., Plagemann, C., & Burgard, W. (2009). Modeling rfid signal strength and tag detection for localization and mapping. In *In proc. ieee int. conf. on robotics and automation (icra* (pp. 3160–3165).

Kaemarungsi, K, & Krishnamurthy, P. (2004). Modeling of indoor positioning systems based on location fingerprinting. *Ieee Infocom 2004*, *00*(100), 1012–1022. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1356988

Kalliola, K. (2008). Bringing navigation indoors. Retrieved from http://www.nokia.com/NOKIA_COM_1/Press/Press_Events/The_Way_We_Live_Next_2008/presentations/TWWLN08_Kimmo_Kalliola.pdf

Kelemen, J. (2003). The agent paradigm. *Computing and Informatics*, *22*(6), 513–519.

Landt, J. (2005). The history of rfid. *Ieee Potentials*, *24*(4), 8–11. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1549751

Li, W, Christensen, H. I., Oreback, A, & Chen, D. (2004). An Architecture for Indoor Navigation. In *Ieee international conference on robotics and automation* (Vol. 2, *April*, pp. 1783–1788). IEEE. IEEE; 1999. Retrieved from IEEE: http://www.cc.gatech.edu/~hic/hic-papers/li-icra2004.pdf

Liu, J., Chen, R., Pei, L., Chen, W., Tenhunen, T., Kuusniemi, H., ... Chen, Y. (2010). Accelerometer assisted robust wireless signal positioning based on a hidden markov model. (pp. 488 –497). doi:10.1109/PLANS.2010.5507251

Morère, Y., & Pruski, A. (2004). A multi-agent control structure for intelligent wheelchair and Aided navigation for disabled people. In *Proceedings of the irs 2004*.

Muñoz Salinas, R., Eugenio, A., Garcia-Silvente, M., & Gómez, M. (2005). A multi-agent system architecture for mobile robot navigation based on fuzzy and visual behaviour. Retrieved from http://journals.cambridge.org/action/displayAbstract?fromPage=online\&aid=350921

NSNotification class reference. (n.d.). Retrieved from http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSNotification_Class/Reference/Reference.html

Ofstad, A., Nicholas, E., Szcodronski, R., & Choudhury, R. R. (2008). 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* (pp. 13–18). San Francisco, California, USA: ACM. doi:http://doi.acm.org/10.1145/1410012.1410016

Ojeda, L., & Borenstein, J. (2006). Non-gps navigation for emergency responders. *Robotics*, 12–15. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.7732&rep=rep1&type=pdf

Oxford English Dictionnary. (2010). Retrieved from http://www.oed.com/view/Entry/182448?redirectedFrom=smartphone#eid117447432

Parnandi, A., Le, K., Vaghela, P., Kolli, A., Dantu, K., Poduri, S., & Sukhatme, G. S. (2010). Coarse in-building localization with smartphones. In *Mobile computing, applications, and services* (Vol. 35, pp. 343–354). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg.

Shanklin, T., Loulier, B., & Matson, E. T. (2011). Embedded Sensors for Indoor Positioning. In *In proceedings of 2011 ieee sensors applications symposium (sas)*.

Silva, P., Paralta, M., Caldeirinha, R., Rodrigues, J., & Serodio, C. (2009). Traceme - indoor real-time location system. (pp. 2721 –2725). doi:10.1109/IECON.2009.5415425

Skog, I, Händel, P, Nilsson, J, & Rantakokko, J. (2010). Zero-velocity detection — an algorithm evaluation. *IEEE Transactions on Biomedical Engineering*, *57*(11), 2657–2666. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/20667801

Specification, T. (2006). Nfc data exchange format (ndef) technical specification. *History*.

Steinmeier, S. (2008). Nfc market update and technology overview. *Agenda*, 1–10.

Want, R. (2006). An introduction to rfid technology. *Pervasive Computing, IEEE*, *5*(1), 25 –33. doi:10.1109/MPRV.2006.2

Want, R., Hopper, A., Falcão, V., & Gibbons, J. (1992). The active badge location system. *ACM Transactions on Information Systems*, *10*(1), 91–102. Retrieved from http://portal.acm.org/citation.cfm?doid=128756.128759

Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press. Retrieved from http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0262731312

Woodman, O. J. (2007). An introduction to inertial navigation.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. *Knowledge Engineering Review*, *10*, 115–152.

Yelamarthi, K., Haas, D., Nielsen, D., & Mothersell, S. (2010). Rfid and gps integrated navigation system for the visually impaired. (pp. 1149 –1152). doi:10.1109/MWSCAS.2010.5548863

Yu, E. (2001). Agent orientation as a modelling paradigm. *Wirtschaftsinformatik*, *43*(2), 123–132.

Zhongshen, L., Low, B., & Filter, P. (2007). Design and analysis of improved butterworth low pass filter. *2007 8th International Conference on Electronic Measurement and Instruments*, 1–729–1–732. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4350554

LIST OF FIGURES

Figure Page