ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Inertial Navigation Systems and Indoor Positioning»

ΕΝΑΛΛΑΚΤΙΚΑ

«Αδρανειακά Συστήματα Πλοήγησης και Εντοπισμός Θέσης σε εσωτερικούς χώρους»

Κουτσαυτίκης Δημήτριος

e-mail : dikoutsa@gmail.com

Επιβλέπων : Βάβαλης Εμμανουήλ

Επιτροπή : Λάλης Σπυρίδων – Γεράσιμος, Κατσαρός Δημήτριος

Βόλος , Ιούλιος 2012

(This page was intentionally left blank)

Contents

Acknowledgements

First of all I would like to thank Professor Manolis Valalis for his insight, guidance and giving up a great deal of his time to help with my research. I would also like to thank my advisors, Assistant Professor Spyridon Lalis and Lecturer Dimitrios Katsaros for their continuous help and support throughout the project.

Next, I wish to thank my mother Penelope, for her support, moral and material, patience and guidance throughout the course of my studies. The largest share of the successful completion of my Master Degree Diploma, I owe it to her.

Many people have helped me to complete the research presented in this thesis. My grandparents, family and friends with their love and support, but especially, my brother and colleague, John, and three friends, I believe that without their support things would be very different. They were always there for me when I needed them. John, Marina, Nikos and Stefanos, thank you.

Eventually, in addition to the individuals named above, the LISyS Lab staff and colleagues as a whole have provided a friendly and vibrant atmosphere in which it was a pleasure to work and have a peace of mind to fulfill this manuscript.

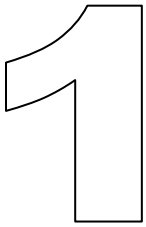
Abstract

The goal of this Master Thesis is to denote the most common practices that are used in Indoor Navigation and highlight the challenges that appear in exploiting Inertial Navigation Systems for achieving accurate indoor positioning. Focus is put on handheld devices and particularly on smartphones, due to the fact that mobile market applications are already designed to use existing digital map databases for localization and guidance procedures. Research of previous work in the field preceded the outcome of this report and pointed out the necessity of developing a new approach that relies mostly on inertial measurements and less in Wi-Fi “signal scans” to determine user position.

Περίληψη

Ο στόχος αυτής της μεταπτυχιακής διατριβής είναι να δείξει τις πιο συνηθισμένες πρακτικές που χρησιμοποιούνται στον τομέα της πλοήγησης σε εσωτερικούς χώρους και να υπογραμμίσει τις προκλήσεις που εμφανίζονται στην αξιοποίηση συστημάτων αδρανειακής πλοήγησης για την επίτευξη ακριβούς εντοπισμού θέσης εντός των κτηρίων. Ιδιαίτερη έμφαση δίνεται στις φορητές συσκευές και ιδιαίτερα στα ευφυή κινητά, γεγονός που οφείλεται στο ότι οι αγορές εφαρμογών για κινητά τηλέφωνα έχουν ήδη σχεδιαστεί ώστε να χρησιμοποιούν τις υπάρχουσες βάσεις δεδομένων ψηφιακών χαρτών για τον εντοπισμό της θέσης των χρηστών και την καθοδήγησή τους. Η έρευνα των υφισταμένων εργασιών στον τομέα προηγήθηκε από την έκβαση αυτής της διατριβής και τόνισε την ανάγκη ανάπτυξης μιας νέας προσέγγισης που βασίζεται ως επί το πλείστον σε μετρήσεις αδρανειακών αισθητήρων και λιγότερο σε «σαρώσεις» Wi-Fi σημάτων για να καθοριστεί η θέση των χρηστών.

1 Introduction – Motivation



1.1 The Indoor Navigation context

In this report, when referring to Indoor Navigation we imply way finding and positioning, within buildings and underground structures, using an **indoor positioning system (IPS).** An IPS relies on nearby anchors (nodes with a known position), which either actively locate tags or provide environmental context for devices to sense, to locate objects or people[1].

First of all, we have to clarify that indoor positioning is not a modern issue. Some traces of previous efforts that focus on localization and guidance procedures, originate back in 1960's[2]. Hence, one may wonder why designing of indoor navigation systems is still a worth looking case nowadays. The answer is quite simple. The problem is not yet solved.

Suddenly, great indoor applications are now being built and becoming really useful to most of us. Spots like museums, stations, airports and malls have already launched them and people use them in everyday life to navigate and access location based content. Indoor Wi-Fi positioning technology isn't a phantasy scenario under investigation anymore.

But as Nick Farina[3] claims: "*Your phone can get you to the museum, but it can't guide you to the T-Rex.*" The underlying truth to the above quote is pretty much significant: Indoor navigation is hard.

[1] http://en.wikipedia.org/wiki/Indoor_positioning_system
[2] http://www.google.com/patents/US7991576
[3] http://radar.oreilly.com/nickf/index.html

Figure 1: Indoor Navigation application on a smartphone[4]

1.2 The methods, the procedures and their properties

The localized nature of an IPS has resulted in design fragmentation, with systems making use of various optical, radio, or even acoustic technologies[5]. Therefore, existing implementations have major drawbacks:

- Cost of preinstalled facilities
- Difficulty in deployment
- Signal interferences
- Technology – specific limitations

What is more, navigation inside buildings features some serious challenges. For one thing, it turns out that creating a useful indoor navigation application requires more than navigation. Therefore, a handy mobile user interface should be more of a "smart guide" and less of a "paper map" on a

[4] http://www.ipms.fraunhofer.de/en/news/press/2011/2011-12-21.html
[5] http://en.wikipedia.org/wiki/Indoor_positioning_system

user's smartphone. One should also involve dynamic data in his application to make it really helpful for the user.

Nevertheless, the big thing is getting and/or tracking user's current location. As you may know, GPS[6] "signal scans" fail on indoors, since satellites' signals are too weak to penetrate even the metal roof of a common car. As a result, existing solutions are provided with a rough initial guess of the true position and afterwards they calculate an estimate of the true, absolute position using only relative position information from motion sensors and digital map information. These map-aided positioning systems can therefore run without GPS support and still yield similar or better functionality (as GPS).

Moreover, another hitch of all previous approaches that are mentioned is that they rely on somehow web-connected user/client devices. This prerequisite is the major feature that prevents us from declaring these systems as autonomous, since the dependencies with the nearby anchors (nodes with a known position) are tightly coupled. In addition to this, designing such systems for inaccuracy is highly inconsistent. Precision depends on signal fluctuations, which depend on factors like how many people are in the room, how they're holding their phone, and other vagaries.

No sooner than this point, one would consider that reaching a satisfying (same or even better than GPS) precision of user's location indoors is a lost case. Unavoidably, construction materials cause the weakening of the satellite signal and that means that it loses significant power indoors, affecting the required coverage for the receivers. Another point is that because of the multiple reflections at surfaces, multiple routes of transmission are caused and therefore uncontrollable errors. Exactly those effects are responsible for the degradation of all known solutions for indoor locating that require electromagnetic waves, from indoor transmitters to indoor receivers.

So a lot of mathematical and physical methods are being used in order to assist us in managing with these problems. Thus, we would say that if systems design shall take into account empirical methods and mathematical modeling as long as inertial measurements, the designers can get back in the game.

"An **inertial navigation system (INS)** is a navigation aid that uses a computer, motion sensors (accelerometers) and rotation sensors (gyroscopes) to continuously calculate via dead reckoning

[6] http://en.wikipedia.org/wiki/Global_Positioning_System

the position, orientation, and velocity (direction and speed of movement) of a moving object without the need for external references. It is used on vehicles such as ships, aircraft, submarines, guided missiles, and spacecraft. Other terms used to refer to inertial navigation systems or closely related devices include inertial guidance system, inertial reference platform, inertial instrument, inertial measurement unit (IMU) and many other variations"[7].

Some pedestrian positioning methods measure steps indirectly in order to determine location. To accomplish that, they suggest that the pedestrian should carry an inertial measurement unit in order to allow the system to perform steps counting. Other solutions propose that the IMU should be foot mounted in order to achieve better precision. In either case, map aid and supplementary sensors are used to restrain the accumulation of subsequent errors that increase over time during inertial navigation. In general, inertial measurements involve differentials of motion and therefore the current position can be determined by integrating, which means that integration constants are required in order to acquire accurate results.

While serious research is being conducted regarding body-mounted sensors and inertial sensors for navigation, guidance and control of vehicles, other projects have already concluded in working solutions in specific fields. The lack of a global perspective and the limitations of application context of these approaches however, have urged us to seek for a less elaborate and a more commonplace counterclaim deployment target: "Smartphones".

1.3 Why smartphones?

One could start by stating the vast expanse of worldwide mobile markets, not only for devices but also for applications, to indicate the essentiality of elaborating a consistent model for indoor navigation. However, to start from the top, it is the nature of an indoor positioning system itself (quick availability, autonomy as well as integration of all system components in an everyday life), that makes modern phones a prerequisite deployment device. They do not require any preinstalled facilities, they can run completely autonomous and all the necessary components (inertial sensors, digital maps) are standard equipment of them.

[7] http://en.wikipedia.org/wiki/Inertial_navigation_system

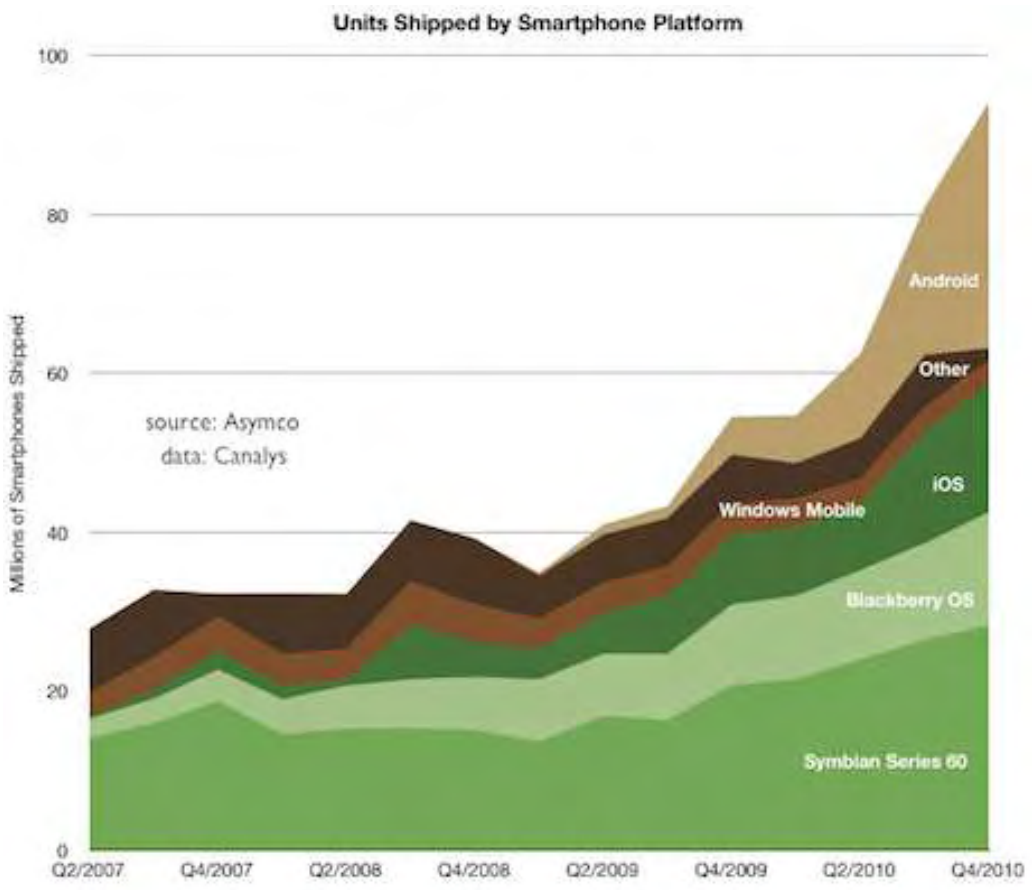


Figure 2: Smartphone devices shipped from second quarter of 2007 to fourth quarter of 2010 of mobile suppliers[8]

To be more accurate, we highlight the lead features that, in our opinion, make smartphones the current best deployment target for an indoor navigation system:

- Device low costs
- Social maps integration
- Straightforward implementation (developer documentation)
- Inertial sensors built in
- Easy distribution (application markets)
- Availability (everyday life gadgets)
- Wi-Fi built in
- Intuitive interface (common user interface patterns)

[8] http://www.guardian.co.uk/technology/blog/2011/feb/09/nokia-burning-platform-memo-elop

Apart from the mobile industry, the whole navigation ecosystem is changing rapidly and in terms of businesses and corporations, navigation market is vastly growing. Since now the ecosystem is completely redefined. An architect with CAD drawings is now a map provider. Facilities can be easily transfigured into navigable sites. Every big retailer willing to drive customers' attention to specific products is a potential INS customer on each own (retailers can now provide guidance through the sales isles if they want). Eventually, every shopping firm, organization, association along with every convention center is wannabe customers.

But while indoor positioning seems to be a "gold rush"[9] for business and corporations, a great paradox arises. In mobile world there are not necessarily navigation giants and certainly there is not a specific leader. By devoting time in reading marketing documents and referring to details provided by most of the IPS vendors, interested customers shouldn't be very lucky in tracing specific aspects on accuracy, reproducibility or any other quality terms. Furthermore, many suppliers do not include a slightest comment to the term "accuracy" in their technical specifications.

Taking all those points in consideration, apparently one can see that developing an indoor navigation system, based on inertial measurements, is a design challenge and that was our main motivation for writing this report, as the possible applications of such system are numerous, much needed and can be immediately deployed in either business or science fields.

Here, we organize this report as followed. In the next chapter we cover the theoretical background by defining inertial sensors and their uses in inertial guidance, we track down inertial navigation's traces through today and eventually we define the field's context in mobile devices describing the current working framework.

In chapter 3 we provide the design principles, extracted throughout our review in the field, of a mobile phone inertial navigation system. The chapter is structured in three parts:

- System Requirements
- Measuring Distance
- Motion sensors overview
- Map representations
- Dealing with inaccuracy

[9] http://bdnooz.com/2009/11/08/indoor-navigation-–-the-new-gold-rush-part-1/#axzz1x9EBt2W4

The preceding chapter of the final includes four system implementations and evaluation analysis of them. Each of these systems stands for a different approach of indoor positioning with smartphones. The first is the outcome of the Bachelor thesis, submitted to Worcester Polytechnic Institute by three students, and it represents a hybrid model based on Wi-Fi signals and inertial measurements to determine location. The second and the third are the products of indoor positioning service providers, Qubulus[10] and Wifarer[11], which rely only on Wi-Fi "signal scans". The last, incomplete though, is the sum of our efforts to develop a system based exclusively on inertial sensors. Eventually, in the fifth and final chapter we state our conclusions and future research analysis.

[10] http://www.qubulus.com/
[11] http://www.wifarer.com/

2 Framework – Theoretical background

2.1 Inertial guidance origins

The term is used to describe a model of a guidance system that can be found in aircrafts, spacecrafts, submarines and even guided missiles. Moreover, corresponding to measurements provided by means of mechanisms, a mounted inertial guidance system automatically adjusts the vehicle's heading, in way appropriate for following an exact given path.

Typically, the system integrates accelerometers and gyroscopes. The common practice for accelerometers is to isolate them from vehicle's angular displacement, by mounting them on gyrostabilized platforms (gimbals). In this way, during periods of vehicle's velocity rate change, the sensors can track changes in motion and gravitation fields by measuring the forces that are needed to sustain their mass at rest, with respect to the movement of the mounting platform. Afterwards, they integrate the data obtained with existing knowledge of position and velocity and alert the control devices to reroute and/or reorientate the vehicle to the corresponding altitude, direction, etc. Gyroscopes on the other hand, are responsible for providing reference directions to accelerometers, thus detecting orientation. Overall, a processor unit computes velocities or directions from sensor readings and compares these data with stored ones. A feedback method called Schuler tuning[12], that accounts for the curvature of the Earth with respect to three axes pointing 'north', 'east', and 'down', is used to improve the system's accuracy.

While inertial navigation seems to be a modern issue, its traces originate back in 1942 in a guidance system used by the Germans V-2 missile [1]. Apart from that, Léon Foucault[13] is credited with naming gyroscope, although he didn't invent it, in 1852. It was actually a free swinging pendulum that rotated with a period that depended on the latitude of its location. Then, in 1908, Schuler[14] developed gyrocompass, but the former device could be only used as a measuring instrument and the latter, although of inertial quality, was only a partial inertial system. In late 1920's and early 1930's the American company Sperry Flight Systems[15] had the vision to optimize the gyrocompass, but the instruments it created during that time were more

[12] http://en.wikipedia.org/wiki/Schuler_tuning
[13] http://en.wikipedia.org/wiki/L%C3%A9on_Foucault
[14] http://en.wikipedia.org/wiki/Max_Schuler
[15] http://www.sperryflight.com/History.html

attitude-indicating (airplane stabilizers, gyrostabilized bomb sights) and less position or velocity indicating devices.

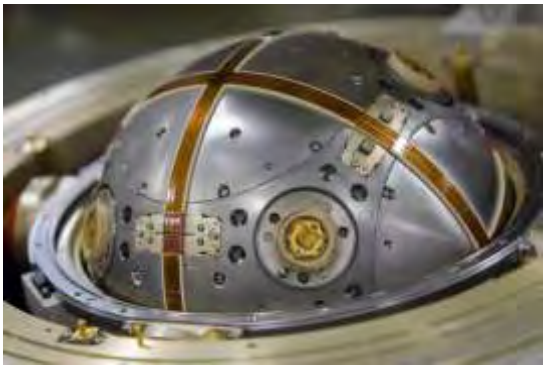


Figure 3: A gyroscopic and inertial guidance module from a nuclear missile[16]

In the late 1940's and early 1950's, M.I.T. Instrumentation Laboratory in cooperation with Northrop[17] and Autonetics[18], aircraft and aerospace manufacture companies, under the sponsorship of American Air Force, began showing deep, purposeful and sincere intent for developing inertial navigation systems. Their efforts resulted in inertial guidance systems for ballistic missiles that could be launched from both land and ships. Eventually, the Space Race[19] between the Soviet Union (USSR) and the United States (US) in 1960's, lead to the optimization of inertial guidance in Apollo[20] spacecraft. The side products of this competition were inertial guidance systems that were integrated into military first and then commercial airplanes. Besides system development, there was simultaneous necessary theory development and subsystem and component benchmarking.

2.2 Inertial sensors

2.2.1 Gyroscope

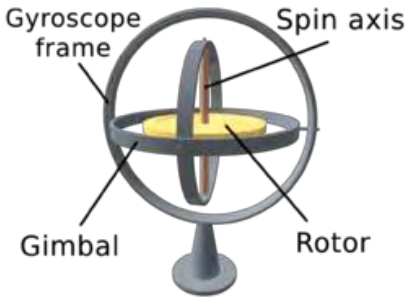
A gyroscope is a rotation sensor capable of measuring or maintaining orientation, based on the principles of angular momentum[21]. Naturally, a mechanical gyroscope is a spinning wheel or disk that spins as the rate of change of angular displacement. The disk is not fixed and it can take any

[16] http://prometheus.med.utah.edu/~bwjones/2009/01/photowalking-utah-hill-afb-museum/
[17] http://en.wikipedia.org/wiki/Northrop
[18] http://en.wikipedia.org/wiki/Autonetics
[19] http://en.wikipedia.org/wiki/Space_Race
[20] http://en.wikipedia.org/wiki/Apollo_program
[21] http://en.wikipedia.org/wiki/Gyroscope

orientation change around its axle. A typical way to measure orientation change is the following scenario.

Gyroscope's disk recognizes the effect of applying an external torque to it, not much in degree but significantly less and in opposite direction of the torque needed to spin it in high rate and overcome the inertia, as orientation change. The best practice, for retaining the orientation of the disk is by minimizing external torque by mounting it in gimbals[22]. Therefore, gyroscopes do not sense rotation in regards to an external reference, but actually the rotation of the platform they are mounted.

Single- and multi-axis models of gyroscopes are available to detect change of angular displacement. The most common multi-axis gyroscope in smartphones is the 3-axis, capable of measuring orientation change in Tait–Bryan angles. While proper Euler angles are equivalent to three combined rotations repeating exactly one axis, Tait–Bryan angles are equivalent to three composed rotations in different axes[23]. These three angles are normally called "Heading, Elevation and Bank", or "Yaw, Pitch and Roll". The later terms are also the names for the three aircraft principal axes.



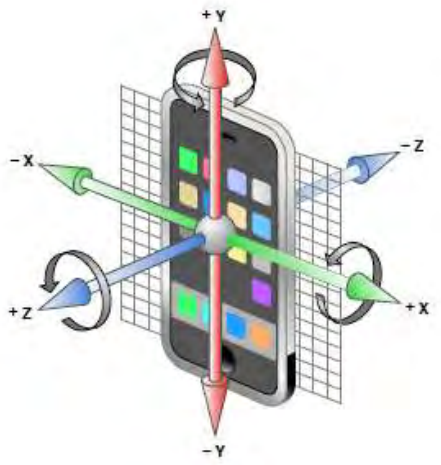


Figure 4: A Mechanical Gyroscope representation[24]

Figure 5: X, Y, Z axis stand for roll, yaw and pitch axis orientation[25]

Although there are many types of gyroscopes, based on other principles, in this report, we cover vibrating Micro Electro-Mechanical System (MEMS) gyroscope[26] devices, found in modern

[22] http://en.wikipedia.org/wiki/Gimbal
[23] http://en.wikipedia.org/wiki/Euler_angles
[24] http://en.wikipedia.org/wiki/Gyroscope
[25] http://developer.apple.com/library/ios/documentation/eventhandling/conceptual/eventhandlingiphoneos/eventhandlingiphoneos.pdf
[26] http://en.wikipedia.org/wiki/Vibrating_structure_gyroscope

mobile phones. Their operation is based on the Coriolis Effect[27], which is generally happening when we have a mass moving and the reference frame is rotating. Focusing on our planet, the Earth's rotation takes one day for a full spin and as a result the Coriolis force is quite small. Therefore, its effects generally become noticeable only for motions occurring over large distances and long periods of time, such as movement of large-scale of weather phenomena, hurricanes for instance.

Back in mobile context, MEMS gyroscopes are implemented as relative high frequency micro-oscillators which force a mass substance to a pendulous motion and measure its movements around and within the pendulant area, with capacitive sense cones. In practice, gyroscopes enable much more responsive user interfaces by allowing various motions to control action and recognizing detailed gestures.

"A pendulous oscillating gyroscopic accelerometer comprising an unbalanced pendulous mass, pivotable on an output axis, that is oscillated about a reference axis transverse to the output axis. The pendulous mass is also oscillated about an input axis transverse to the reference axis, and the acceleration force along the input axis is determined from at least one of those oscillations."[28]

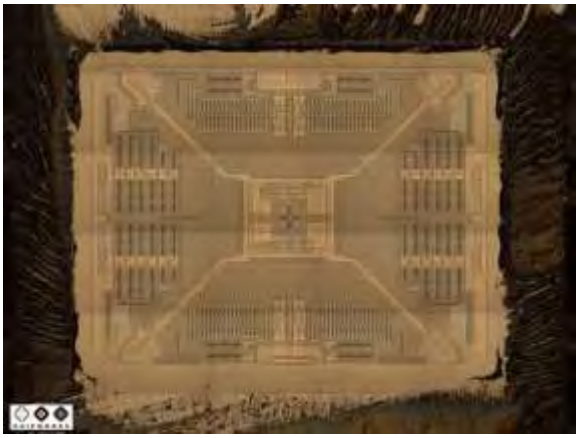


Figure 6: iPhone MEMS gyroscope[29]

[27] http://en.wikipedia.org/wiki/Coriolis_effect
[28] http://www.google.com/patents/US5457993
[29] http://www.chipworks.com/

2.2.2 Accelerometer

An accelerometer is a device that measures proper acceleration, also called the four-acceleration. This is the acceleration "felt" actually by people and objects and it is also the same acceleration the sensor itself would experience in a free fall reference frame (inertial reference frame).

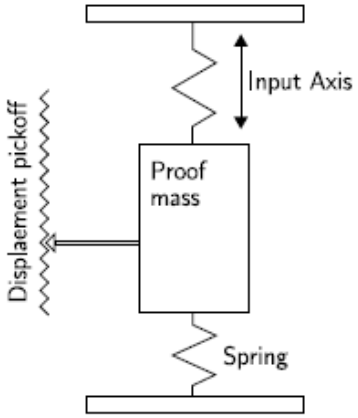
"According to the last statement, an accelerometer measures weight per unit of mass, a quantity with dimensions of acceleration that is sometimes known as specific force, or g-force (although it is not a force). Acceleration is quantified in the SI unit meters per second per second (m/s^2) or popularly in terms of g-force (g)"[30].

In other words, if a local inertial frame exists at any point in space and time, then measured acceleration is the one relative to that frame. Most commonly, these measurements are taken in terms of g-force. Consequently, due to the fact that all points in Earth's surface are accelerating upwards relative to the local inertial frame (frame of a free falling object near the surface), any acceleration sensor that is put at rest on our planet's ground will indicate approximately 1 g upwards.

Thus, to obtain proper readings on output, which is critical for use in an inertial navigation system, this "gravity offset" must be subtracted and corrections for effects caused by the Earth's rotation relative to the inertial frame may also be needed. In practice, this can be achieved by calibrating the device at rest or from a known model of approximate local gravity in current sensing position.

Conceptually, an accelerometer can be visualized as a mass on a spring [2]. The measurable acceleration is the displacement occurred when the mass is displaced in way that the spring accelerates the mass with the same rate as the mounted platform. In plain words, an accelerometer picks up any kind of movements of the platform the spring attached and thus it also behaves as motion sensor, but the hitch is that the sensor itself is extremely sensitive and it cannot give deterministic results to help someone fathom the actual motion that exhibited certain readings on output.

[30] http://en.wikipedia.org/wiki/Accelerometer



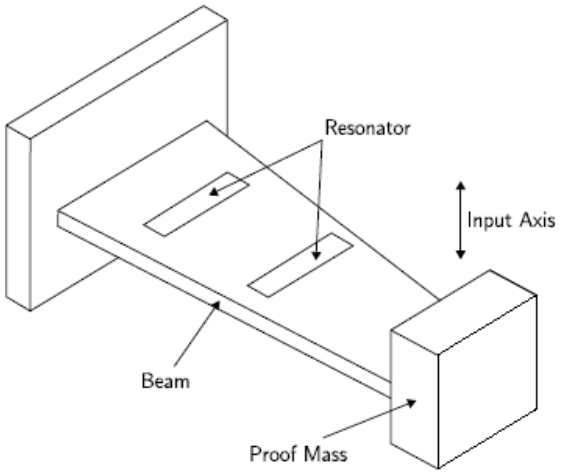


Figure 7: A mechanical accelerometer[31]

Figure 8: A MEMS accelerometer[32]

Nowadays, industry distributes a great variety of accelerometers in worldwide markets. Most of them do not directly output the measured values to a display, but provide them to other devices. These sensors also have practical limitations in how quickly they respond to changes in rate of velocity, and cannot respond to changes above a certain frequency of change.

What is more, single- and multi-axis models of accelerometer are available and while in most projects 2-axis is enough, in an inertial navigation system and particularly in 3d positioning, a 3-axis or a 2-axis, mounted at right angles, accelerometer is needed. For this reason, 3-axis micro machined (MEMS) accelerometers are most common in latest mobile devices. Generally, their presence in smartphones indicates the ability to detect position, change in orientation and provide game input.

2.2.3 Sensor fusion

Prior to inertial sensors which equip modern mobile phones, magnetic field sensors (compasses[33]) are also significant in the scope of this report. Nevertheless, using gyroscope measurements, instead of compass ones, is proved to lead to much more consistent solution.

[31] http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf
[32] http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf
[33] http://en.wikipedia.org/wiki/MEMS_magnetic_field_sensor

This statement arises from the fact that compasses are sensitive to magnetic field interferences and depend on:

- Magnetic declination[34]
- Magnetic inclination[35]
- External references (Earth's Poles)

Magnetic and inertial sensors, in combination, are able to provide accurate motion estimates and are well suited for use in many navigation tasks. Here, we have to assert that distinguishing software fusion, from hardware is a difficult or even impossible task. In most cases they are tightly bound during implementation and design. The most literal case of hardware blending seems to be Gyrocompass[36], however it is not applicable on mobile devices. Another conventional term is 6-axis accelerometer, owing to the fact that such a sensor does not have actually 6-axis but it is rather the result of blending a 3-axis gyroscope, 3-axis accelerometer together with an onboard digital motion processor capable of processing complex motion fusion algorithms.

Putting together with the native physical and hardware limitations of each sensor alone, apparently it seems that correct data fusion and thus overall performance is close related to initial calibration and accurate transforming algorithms between sensors [3]. Not only the calibration information is generally hard to obtain and time-consuming, but also normally requires additional equipment.

The integration result of magnetic and inertial sensors into one device is popularly known as an Inertial Measurement Unit[37] (IMU). Since now, the IMU integrations in smartphones have concluded in successful applications in the fields of:

- User Interface
- Gaming
- Image Stabilization
- Navigation
- Augmented Reality[38]

[34] http://en.wikipedia.org/wiki/Magnetic_declination
[35] http://en.wikipedia.org/wiki/Magnetic_dip
[36] http://en.wikipedia.org/wiki/Gyrocompass
[37] http://en.wikipedia.org/wiki/Inertial_measurement_unit

Sensor	100Hz		20Hz	
	Total	Application	Total	Application
Sensor fusion	65%	20%	65%	10%
Accelerometer	50%	15%	46%	5%
Accel + gyro	51%	10%	50%	5%

Table 1: Total CPU usage for an application running on iPhone 4.0, Credit: Jeffrey Powers, Occipital

2.3 The mobile phones context

In our days, wireless networks are met in almost every indoor place and since all modern technology smartphones have Wi-Fi capable of scanning for known access points built in, the user's position could be easily calculated with the usage of trilateration[39]. Nevertheless, due to the inefficiencies of indoor Wi-Fi positioning technics and technologies described in the early start of this report, we put the center of our attention on indoor navigation based on inertial sensors.

Above all manufacturers and platforms, we focus on Google's Android and Apple's iOS (iPhone, iPad, iPod Touch) mobile devices due to their rapid increasing markets, their good documentation and potential of integrating new technology faster than other platforms. However, in the early beginning, we trace some significant variations in application development policy issues between these two vendors. But before that, let's revise the platforms' feature release history.

Features	Android		iOS	
	Date	Device	Date	Device
Accelerometer	October 22, 2008	HTC Dream[40]	January 9, 2007	iphone(original)[41]
Gyroscope	November 15, 2010	Nexus S[42]	June 7, 2010	iphone 4[43]
SDK	August 18, 2008[44]	-	March 6, 2008[45]	-
App Store	22 October 2008[46]	-	July 10, 2008[47]	-

Table 2: Android and iOS first feature release dates

[38] http://en.wikipedia.org/wiki/Augmented_reality
[39] http://en.wikipedia.org/wiki/Trilateration
[40] http://en.wikipedia.org/wiki/HTC_Dream
[41] http://en.wikipedia.org/wiki/IPhone_(original)
[42] http://en.wikipedia.org/wiki/Nexus_S
[43] http://en.wikipedia.org/wiki/IPhone_4
[44] http://en.wikipedia.org/wiki/Android_software_development
[45] http://en.wikipedia.org/wiki/IOS
[46] http://en.wikipedia.org/wiki/Google_Play
[47] http://en.wikipedia.org/wiki/App_Store_(iOS)

Taking a closer look and evaluating the above notes, we would say that Apple appears to have a head start in the scope of integrating inertial sensors and providing tools not only to use them in applications but also distributing such software services to mobile owners easily. Although, Apple's devices seem to win the race of first come innovation, Google's Android phones seem able to follow up or even bridge the gap.

Back in inertial navigation, some projects, described at Chapter 4, include Wi-Fi positioning within buildings, aided by inertial guidance measurements. At this point, we have to highlight a catch in using Wi-Fi signals for indoor positioning. Although Android's developers are allowed to perform Wi-Fi signal scans, iPhone developers do not have the same privileges. Of course there are alternative methods, which use the buildings in order to perform the tracking instead of the device. For instance, Cisco MSE[48] and some other Wi-Fi installations are able to detect indoor locations of any wireless device in their coverage area.

Another case is that Google's Maps for indoors are available only for Android[49] smartphones. This seems like a major hitch for using easily integrated and social maps in iOS devices. But while Google starts putting limits, charging extended use of Maps API, Apple has announced the use of new maps application[50]. To confirm once again the rapid changes in mobile industry, here we note that the later happened during the time we build this report, so we revised this entry.

Overall, one cannot argue that both Android and iOS applications are close related in terms of usability, at least in most cases. But while Apple's hardware is of really high quality, in the Android world, one can buy from very cheap and poorly designed smartphones, to fully equipped (with all existing inertial sensors and top of the line hardware features) ones, with enhanced user experience. In terms of indoor positioning, with the right hardware(inertial sensors) and software combination(digital maps), both of these mobile operating systems can shine, and it appears that a system designer is able to choose one of his personal preference, without worrying about a huge gap between the two platforms.

[48] http://www.cisco.com/en/US/products/ps9742/index.html
[49] http://support.google.com/gmm/bin/answer.py?hl=en&answer=1685827&topic=1685871&ctx=topic
[50] http://www.apple.com/ios/ios6/maps/

To conclude the theoretical background and current framework analysis, in the next sections, we highlight the existing working methodologies for locating user's indoor position in smartphones:

- Wi-Fi tracking
- Dead reckoning
- Map-matching

2.4 Current position techniques

2.4.1 Wi-Fi tracking

Despite the already mentioned inefficiencies of location determination inside structures, using Wi-Fi "signal scanning" methods, we cannot argue that these techniques are valid candidates in the field of indoor positioning. Hence, we provide honorable notes of the most common algorithmic approaches in this section.

There are various, algorithmic based, solutions to obtain a user's location and they reach several degrees of precision and accuracy, depending on the application. For instance, an acceptable scenario may be that the locating target (object or person) is on a certain building floor (degree of precision) and this calculation generates the same result at 95 percent of the execution times (degree of accuracy). On the scope of this report, the general whereabouts of a person are considered as the sufficient desired precision level.

Among several approaches for determining position, the most common are the following[51]:

- Nearest sensor
- Trilateration and triangulation.
- Time distance of arrival (TDOA)
- Received signal strength indication (RSSI)
- GPS
- Fingerprinting
- NAVSOP

[51] http://features.techworld.com/mobile-wireless/2374/all-about-wi-fi-location-tracking/

Figure 9: Wi-Fi signal scanning representation[52]

Next, we take a closer look to the main functionalities of these algorithms.

Nearest sensor

It is the least complex approach, yet with the least precision among the others. Most vendors of wireless networks include this feature in their management systems. Nearest sensor method determines the wireless access point (802.11) or cellular base station to which a client device is associated. Consequently, the algorithm assumes the user's device is most closely connected to that sensor and it calculates the distance of the signal transmission.

This calculated estimation of client-sensor proximity has the precision of the diameter of a full, three dimensional, circled "cell" that corresponds to the sensor's signal transmission area. What is worse, it presumes that the client device is associated with the nearest sensor, despite the fact that this is not actually the case. For example, let's say we apply the method using a wireless access point of about 50-by-50 meters coverage. Then the nearest-sensor associates any client within this coverage area (2.500 square meters) with this sensor, ignoring the fact the client might have already been associated with another sensor that its signal overlaps the area, due to signal attenuation or overload of the former one.

[52] http://gpsnavigationsystem-s.com/gps-navigation-systems/gps-technology/indoor-positioning-system.html

Triangulation/trilateration

The nearest-sensor calculations can be imported to other algorithms in order to determine position more precisely. Triangulation pinpoints location by tracking intersections between three or more nearby sensors (or other anchor points), measuring the angles between them. This method's precision varies from 25 to 100 meters. Trilateration on the other hand, measures the distance between sensors or other anchor points, rather than the angles between them. Trilateration seems to achieve better precision, of about 10 meters.

	Best Accuracy	**Underlying Technology**	**Available on Smartphones**
Wide Signal Strength Fingerprinting[53]	2.48m	GSM	no
Skyhook(GSM)[54]	200m	GSM	yes
Navizon(GSM)[55]	50m	GSM	yes
Skyhook(Wi-Fi)	10m	Wi-Fi	yes
Navizon(Wi-Fi)	20m	Wi-Fi	yes
RADAR[56]	2m	WaveLan	no
GP for Signal Strength-Based Location Estimation[57]	2m	Wi-Fi	yes
Ekahau Positioning Engine[58]	1m	Wi-Fi	no
The Bat[59]	3cm	Ultrasound	no
The Cricket[60]	3cm	Ultrasound	no
Lok8[61]	**Sub-meter**	**Ultrasound**	**yes**

Table 3: Comparison of Indoor Positioning Implementations

[53] http://dspace.utlib.ee/dspace/bitstream/handle/10062/494/otsason.pdf?.
[54] http://www.skyhookwireless.com/location-technology/performance.php
[55] http://www.navizon.com/its/whitepaper.pdf
[56] http://paper.ijcsns.org/07_book/201005/20100518.pdf
[57] http://www.roboticsproceedings.org/rss02/p39.pdf
[58] http://www.ubitel.ru/files/epe%204.1%20user%20guide.pdf
[59] http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/
[60] http://cricket.csail.mit.edu/
[61] http://arrow.dit.ie/cgi/viewcontent.cgi?article=1021&context=ittpapnin

TDOA, RSSI, GPS

Standing mostly on the cellular side, GPS systems use triangulation combined with a metric known as time difference of arrival (TDOA) over a network of satellites. This method, computes relative time delay of signals that arrive at different sensors and can be used with triangulation in wireless networks, apart from the cell ones, too. Time is proportional to the distance traveled, so client's position can be obtained by estimating the distance to each sensor within range. Combined with TDOA measurements, received signal strength indication (RSSI) can be used to measure the radio signal power loss between transmitter and receiver to calculate distance. Here, there is no point in stating GPS accuracy in the aspect of indoor positioning as satellite reception within buildings fails.

Fingerprinting

Location fingerprinting method is a more sophisticated category of position tracking that observes the operating environmental parameters, such as moving people, furniture, windows, attenuation and multipath interferences on the wireless signal. From intelligence gathered at the observation procedure it estimates a mobile device's current location. Under the hood of this algorithm is the assumption, in analogy to the fact that every person has a unique fingerprint, that every physical location has a unique characteristic (fingerprint) in wireless signal space.

A fingerprint of the observed wireless environment is calculated by a physical walk-around the sampling field using a handheld device capable of Wi-Fi scanning (or sniffing). Afterwards, the recorded data are compared to the real-time measurement deviations in order to determine the device position[62]. In general, the fingerprinting procedure consists of two phases: a recording (sampling) phase and a positioning (locating) phase. The first operation includes data collection and raw data filtering together with a "signal" map construction from the recorded dataset. The second procedure involves determining position of the mobile device through exploiting real-time measurements and intelligence gathered from the "signal" map database. In this way, the exact implementation depends heavily on fulfilling the different level of performance requirements used in the recording phase and the scheme used in the positioning phase (deterministic and stochastic approaches for instance).

[62] http://code.google.com/p/wlan-pos/

NAVSOP

The acronym NAVSOP stands for Navigation via Signals of Opportunity[63] and it refers to a new concept in which a system can track position by "listening" to early everything that transmits a signal, whether it is coming from satellite, a wireless access point, a TV or every nearby radio in general. The system does not need to be aware of the signal source in order to geolocate and it has one major advantage. Not only it is robust against signal jamming but it can also benefit from jamming attempts to relocate position more precisely. If both BAE Systems[64] (NAVSOP inventor) and world governments agree to allow civilians to use such technology, there is a strong chance that NAVSOP is deployed on smartphones.

Figure 10: Concept scheme of Navigation via Signals of Opportunity

Wi-Fi tracking system supplier approaches

The industry ecosystem of Wi-Fi tracking approaches for location finding inside buildings is significantly sparse nowadays. While some suppliers, who are responsible for manufacturing wireless LAN communication systems, integrate location-tracking technology as a layer within their systems, other third-party location experts provide such services by making overlays of wireless tracking systems.

[63] http://www.engadget.com/2012/06/30/bae-systems-navsop-can-do-positioning-without-gps/
[64] http://www.baesystems.com/home

Nonetheless, many Wi-Fi system vendors prefer the fingerprinting implementations, using intelligent algorithms to extract knowledge from observing environmental effects on wireless signals, in order to enable security and management of their own wireless LAN networks. The same approach is also followed by a number of third-party location specialists.

2.4.2 Dead reckoning

"In navigation, dead reckoning (also ded for deduced, reckoning or DR) is the process of calculating one's current position by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time, and course"[65].

An inertial navigation system is often described as a very accurate dead reckoning system. In other words, the inertial system carries out course deviation measurements in order to exhibit corrective commands to drive the system from a displaced, according to the desired route, position where it is, to a position where it isn't but should be. Therefore, the system will advance from the current indicated position to the one where it isn't and accordingly, the position where it is, is now the position where it wasn't.

Since each new position is subject to last position estimation, a dead reckoning system is a relative system. Every faulty calculation on the first step of the procedure triggers cumulative errors to other computations. In this way, any error occurred at each phase remains constant and all subsequent errors are cumulative, increasing over time.

The DR system does not rely on GPS, beacons, or landmarks. It is self-contained navigation system, capable of measuring and tracking the momentary location and route trajectory without reference to external information sources. Consequently, DR systems are useful in GPS-denied environments, within structures, underground tunnels, or dense forests, for instance. Military forces, security personnel and emergency responders are potential users of such systems.

[65] http://en.wikipedia.org/wiki/Dead_reckoning

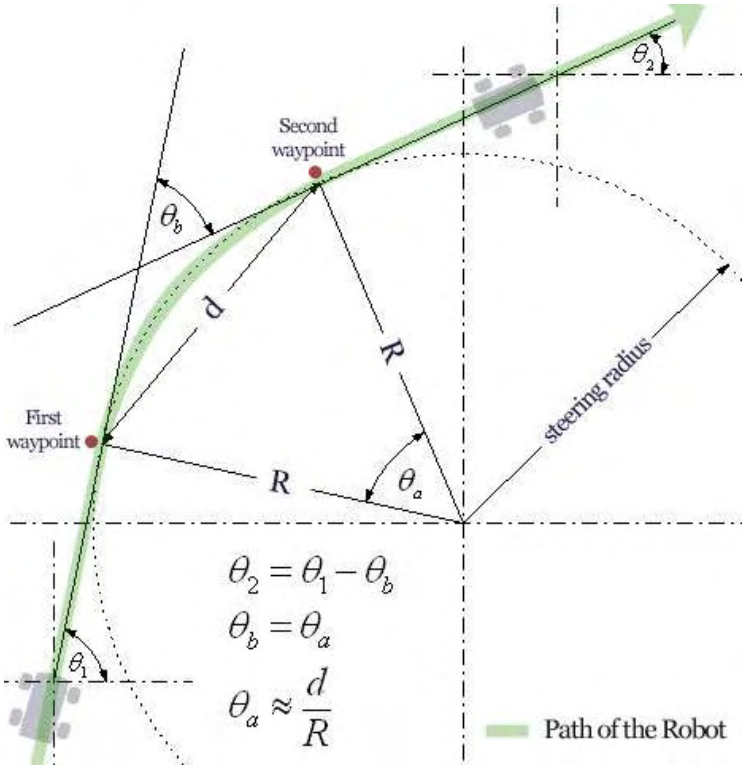


Figure 11: A car dead reckoning navigation[66]

Due to the fact that DR systems are close related to inertial navigation ones we will cover more at chapters 3 and 4, where we describe design principles and present implementations, accordingly.

2.4.3 Map matching

Furthermore, apart from dead reckoning, position tracking estimation is carried out with the aid of computer vision with surprisingly accurate results[4][5]. According to this approach, image scans, such as the ones taken by smartphone cameras, are compared and matched against a consistent and pre-determined dataset. This technique benefits from the fact that the application context can be treated as fixed, without environmental changes, but potentially it requires heavily constructed and pretty large databases. Thus, in this report we do not cover solutions and implementations based on this approach, because global localization and mapping seem to be an "unreachable" goal.

[66] http://www.ikalogic.com/wfr-a-dead-reckoning-robot/

Another candidate in indoor positioning, independent of Wi-Fi tracking or precompiled databases, are the popularly known as Simultaneous Location and Mapping (SLAM) methods [8]. These techniques involve tracking position of autonomous devices (robots, vehicles, even handheld devices) for map construction of unexplored environments (without previous knowledge) or for revising a map model of an explored environment (with previous knowledge gathered from a given map). Here, we focus on the second case.

Strictly, we refer to the term map matching to describe the fusion process of digital map and positioning data. Particularly, in a map matching approach the goal is to merge (inertial or Wi-Fi Tracking) positioning measurements with digital map information in order to determine a client's mobile device location that most accurately matches the digital map. These methods supplement the dead reckoning techniques which are subject to errors. In general, map matching efforts are more efficient in providing guidance on a certain route, a problem proportional to following a vehicle's course using a GPS device.

Several proposed systems have been designed to virtualize environments in the perspective of autonomous indoor navigation. Some are based on accurate geometrical maps constructed from raw data or single lines, where others rely on precise topological maps that use symbolic descriptions. While one approach seems to be optimal reaching some requirements (i.e. global localization), at the same time it can be tedious to achieve accurate results that are based on certain principles (i.e. precise geometric relations). Moreover, map aided efforts for location detection prove more efficient as the accuracy and precision of existing perceptions decrease. Especially, none of the methods seems to minimize the large scale environments consistency problems. The world model consists of a huge variety of large and complex indoor environments (multi-floor structures or groups of structures, with multi-room floors) that makes creating the set of predetermined routes or way finding very inaccurate and inefficient.

What is more, in the terms of determining location within buildings, the final objective is to provide route related data as well as guidance to the desired destination. However, the whole concept of way finding and guidance procedure has to be revised as limitations, such as application specific map models, variety of human displacement within buildings and significant user requirements, occur from the early start. Fortunately, these problems have been alleviated in exploiting combinations of techniques.

Particularly, to compensate for the global convergence and consistency of the built maps in large environments, it is necessary to combine a metric approach along with a topological approach in using several local maps in order to give a global metric description.

The Relative Maps (metric approach)

This method includes calculating distances between the landmark points which are relative to reference known positions. Naturally, the relative map dataset is built by computing metric features (latitude and longitude, elevation, orientation) of all anchor points in relation to a reference point (or so-called starting point). In general, maps provide an approximate virtualization of a state of the runtime environment. Here we note that this representation is likely to be inconsistent to the real world scenario. The fundamental idea behind using relative maps, is introducing a map state where distance is a quantity invariant under the tracking device's (robot, vehicle or handheld device) instant configuration (orientation or heading). Owing to this, we can discriminate between the device position and the landmark estimation and as a consequence the landmark estimation is independent of potential fault (due to inertial measurements) guidance of the device. As soon as a relative map is constructed and the exact position of a group of landmarks is calculated (with the aid of initial calibration), it is possible to match the absolute location of the device to the absolute, real world, map.

The Topological Maps

The topological map is a high level approach, comparing to the metric one, in which distinctive features, such as doors, windows and elevators, within buildings are encoded in order to achieve a compact representation. One of the main problems of this method is the perceptual aliasing, where certain locations inside structures appear identical to the device's sensors. A common case is Z-level or altitude position estimation within multi-floor buildings, in which most efforts fail to detect the exact device's floor location (for instance they assume the device is on the third floor while it is on the second). To compensate for the perceptual aliasing problem, the distinctiveness of the environment must be improved. To achieve this and maximize the reliability in navigation, all available measurements from sensor fusion must be used.

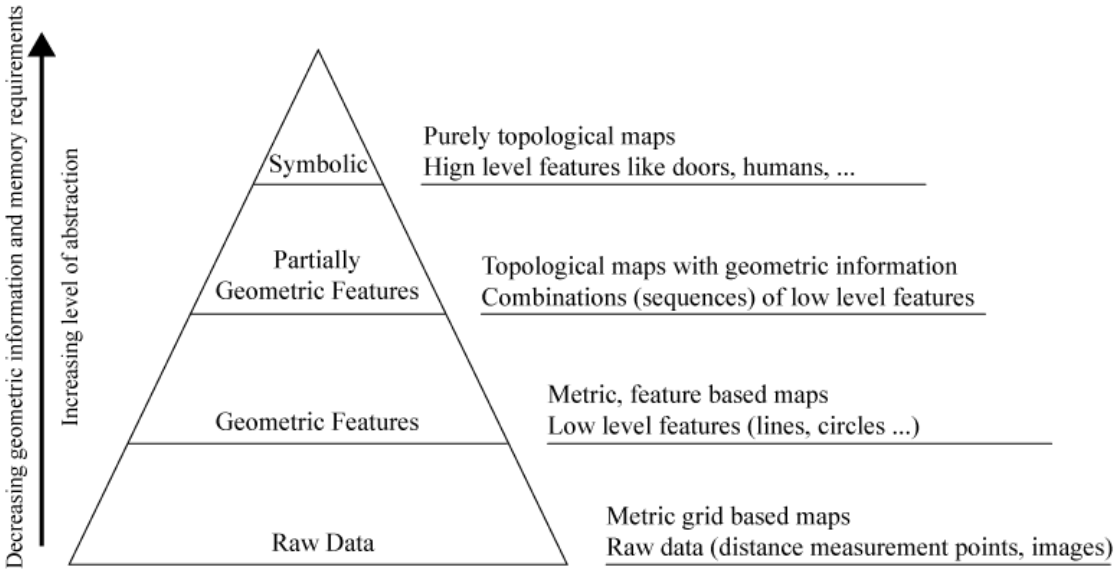


Figure 12: Hierarchy of abstraction levels. As we go up the hierarchy, we increase the distinctiveness and we reduce the geometric information. For global localization and mapping, high distinctiveness is of importance, whereas for local action, precise geometric relations with the environment come forward. [8]

The final result of map-matching algorithms is the coordinates of the point that represents the actual position of the device on the predetermined map model. When a map matching is performed, the device adapts its position to the coordinates of the matching point and then it is in hands of appropriate user interface to provide guidance related or destination estimation data to the system client.

3. Design principles - Dealing with inaccuracy

3.1. System Requirements

In this chapter we provide the desired design principles, extracted throughout our review in the field, of a potential mobile phone inertial navigation system. Particularly, the goal here is to design a self-contained navigation system, capable of measuring and tracking the user's momentary location and route trajectory without reference to external information sources. The development platforms are Android or iOS mobile devices with appropriate hardware (accelerometer and gyroscope) and software features (digital maps, multithreading, background tasking).

The target of this system is to precisely determine the general whereabouts of a person that moves inside a building, without using any Wi-Fi tracking method (of the preceding ones at section 2.4.1), but only with the aid of inertial measurements together with a digital map. At this point, we have to state the general assumptions that emerged while we dug further into defining system requirements. First, we assume that mobile users do not perform running and jogging motions during navigation, but only walk according to normal walking patterns. We also claim that the navigating users do not really know where they are or where they have been. They are only aware of where they entered a place (the whole building, a single floor or room) and this exact position is a potential calibration point. In addition, users can be completely disorientated and head off 180 degrees, in the opposite, of their desired destination, direction during their walk around. Lastly, one major assumption is that perfection is not required. Let's not forget that sufficient precision level is detection of user's general whereabouts inside a building. The system is not required to give them exact output of the landmarks (object and general points of interest with fixed position) users face up into, but it has to be capable of notifying them of nearby ones.

Eventually, the outcome of developing a mobile phone inertial navigation system, will be a distributable (without use of any private APIs) mobile application through Google's or Apple's mobile application markets.

3.2 Determining process hierarchy

Regarding software components integration into one system, capable of tracking user's location by using inertial measurements, we suggest the following scheme:

Figure 13: System Integration

When deciding the processes hierarchy, weighting subcomponents' relative "importance" to the inertial navigation system is a very crucial development issue, in terms of trade-offs in accuracy, speed, and battery-efficiency. Before we start briefing on how Android and iOS manage the processes lifecycle, we have to provide some basic terminology:

- The term **thread** can be defined as a separate path of code execution.
- **Process** refers to a running executable, which may involve multiple threads.
- The term **task** stands for the abstract concept of work that needs to be performed.

When an application is launched, the operating system creates an execution thread for the application, known as "main" thread. This thread is of high importance as it propagates dispatching events (including drawing ones) to appropriate user interface elements (widgets, navigation bars, etc.). Moreover, all components that run in the same process are initialized in the UI thread, and system callbacks (such as button touchedInside or onKeyDown) always run in that thread. Hence, each component instantiation does not involve separate thread creation. In the scope of this report, some implemented methods might be called from multiple threads and for this reason they must be written to be thread-safe.

To determine which processes to keep and which to kill (or suspend), Android or iOS determine each process's "importance hierarchy"[67] focusing on the running components in the process and the state of them. Next we note the common process hierarchy that is traced in both platforms.

[67] http://developer.android.com/guide/components/processes-and-threads.html

Foreground process

A process that is required for current user actions. Particularly, a process is considered to be in the foreground if it hosts an activity that the user is interacting with. In most cases, simultaneous foreground processes are limited at any given time.

Visible process

A process that does not have any foreground components, but still can affect the current user interface. A visible process is considered extremely important and will not be killed unless the system cannot maintain all foreground processes execution and still retain sufficient memory levels.

Background process

A process which supports a seamless activity, that is not currently visible to the user. These processes do not alter the user experience significantly or directly (background music play for instance). Generally, there are many running background processes, placed in an LRU (least recently used) list and they are used to implement activity lifecycle methods, restoring all of the activity visible states, when an application reverts from suspended mode back to active.

Empty process

A process that does not involve any active application component. The system does not kill such type of processes for caching purposes, in order to improve restoring time when an application returns to active mode.

Applying the above rules of process “importance hierarchy" to our design goal, one has to efficiently determine the correct hierarchy scheme of the software components described in Figure 13. For one thing, a developer must compensate for executing an inertial navigation application in an Android or iOS device, both on foreground and background mode, in terms of battery reserving and overall application consistency and performance. Depending on the current mode, the appropriate scheme has to be applied, thus we provide the following figures corresponding to the two execution scenarios.

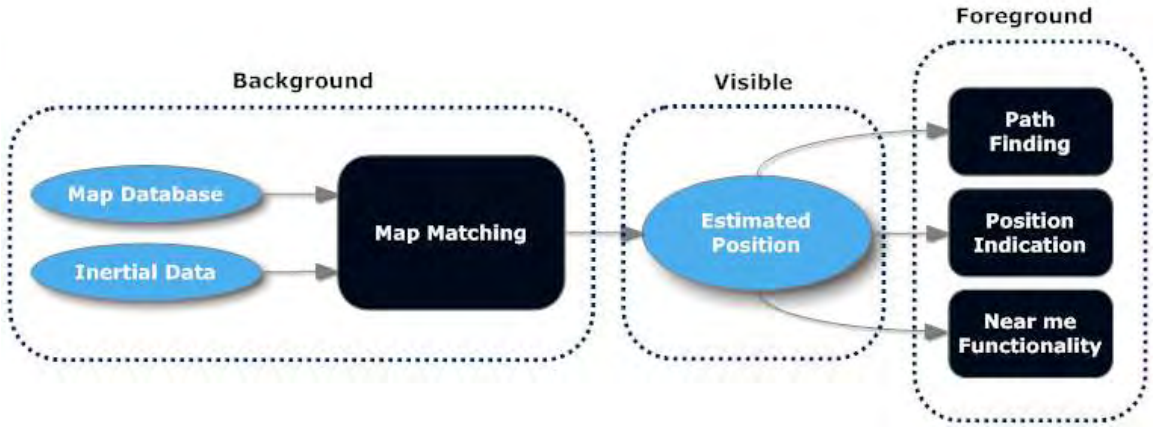


Figure 14: Process hierarchy scheme in foreground mode.

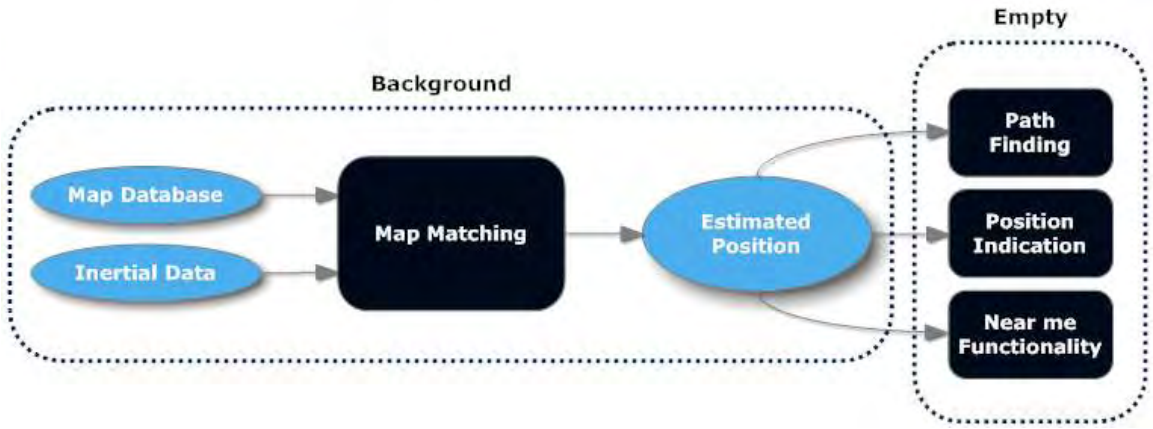


Figure 15: Process hierarchy scheme in background mode.

3.3 Motion Sensors overview

Modern mobile platforms provide several sensors that enable the motion monitoring of a device. Here, we discriminate hardware-based sensors (accelerometer and gyroscope), to the sensors that can be either hardware-based or software-based (gravity, linear acceleration, and rotation vector sensors)[68]. In general, software-based sensors are available in devices that involve one or more hardware sensors.

Naturally, accelerometers and gyroscopes are used for motion sensing, such as bumping, shaking or rotation. There are two types of motion sources. Direct user input (a user performs a shake gesture) and physical environment input in which the device is sitting (for example, moving with you while you drive your car). In the first case, motion detection applies to device's or application's inertial frame of reference while, in the second case you are monitoring estimation

[68] http://developer.android.com/guide/topics/sensors/sensors_motion.html

is relative to the world's frame of reference. Accelerometers alone, are not typically used to determine the mobile phone's position, but combined with other sensors, such gyroscopes or magnetometers, they can estimate the device's position relative to the earth's reference frame.

3.3.1 Isolating the Gravity Component from Accelerometer Data

In cases where accelerometer measurements are user for detection of a device's current orientation, the application should be able to filter out the portion of the acceleration data caused by gravity from the portion of the data that is caused by motion of the device. To achieve this, a low-pass filter can be used in order to reduce the influence of sudden changes on the accelerometer data. The resulting filtered values then reflect the more constant effects of gravity. Because acceleration data comes in regularly, these values settle out quickly and respond slowly to sudden but short-lived changes in motion.

Programmatically, this can be achieved by applying a low-pass filter as described below :

```
// Define a filtering factor
float alpha = 0.8;
// Isolate the force of gravity with the low-pass filter.
acceleration[x]  = alpha * gravity[x] + (1 - alpha) * old.acceleration[x]
```

In this example acceleration[x] is the current raw sensor's acceleration value measured in X axis and old.acceleration[x] is the previously filtered value of acceleration. The parameter alpha is a a high-value filtering factor to generate a value that uses 80 percent of the unfiltered acceleration data and 20 percent of the previously filtered value. Eventually, gravity[x] stands for earth's gravity force in X axis.

3.3.2 Isolating Instantaneous Motion Component from Accelerometer Data

In cases where accelerometers are used to detect just the instant motion of a device, the application should be able to isolate sudden changes in movement from the constant effect of gravity. This concept is also known as implementing a linear accelerometer. The linear acceleration sensor measurements suffer from an offset bias, which has to be removed. One

simple way to achieve this is to ask the user to calibrate the device by setting the device on a table, then read the offsets for all three axes and eventually subtract that offset from accelerometer's raw data to get the actual linear acceleration.

Programmatically, this can be achieved by applying a high-pass filter as described below :

```
// Define a filtering factor
float alpha = 0.8;
// Remove the gravity contribution with the high-pass filter.
linear.acceleration[x]  = acceleration[x] – (alpha * gravity[x] + (1 - alpha) * old.acceleration[x])
```

In this example acceleration[x] is the low-pass filtered, free of gravity force, acceleration value in X axis and old.acceleration[x] is the previously filtered value of acceleration. The parameter alpha is a high-value filtering factor to generate a value that uses 80 percent of the unfiltered acceleration data and 20 percent of the previously filtered value. Eventually, gravity[x] stands for earth's gravity force in X axis.

3.3.3 Using the Rotation Vector Sensor

The rotation vector represents the orientation of the device as a combination of an angle and an axis, in which the device has rotated through an angle θ around an axis (x, y, or z)[69]. The three, unitless, elements of the rotation vector are expressed as x*sin(θ/2), y*sin(θ/2), z*sin(θ/2), accordingly. The x, y, and z axes are defined in the same way as the acceleration sensor. In this way, the magnitude of the rotation vector is equal to sin(θ/2), and the direction of the rotation vector is equal to the direction of the axis of rotation.

This coordinate system we described above has the following characteristics:

- X is defined as the vector product Y x Z. It is tangential to the ground at the device's current location and points approximately to East.
- Y is tangential to the ground at the device's current location and points toward the geomagnetic North Pole.
- Z points toward the sky and is perpendicular to the ground plane.

[69] http://developer.android.com/guide/topics/sensors/sensors_motion.html

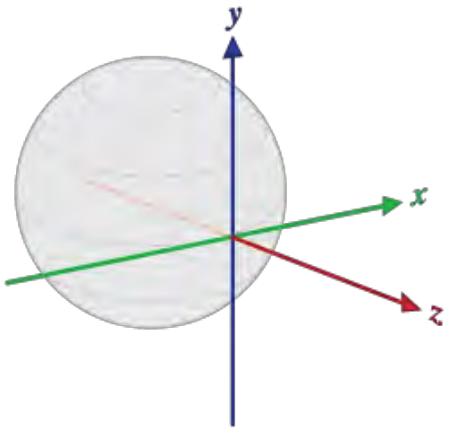


Figure 16: Coordinate system used by the rotation vector sensor[70]

3.3.4 Sensors Coordinate System

In general, the sensor framework uses a standard 3-axis coordinate system to express data values. Almost all MEMS sensors that are found in smartphones define their coordinate in relation to the device's screen when the device is held in portrait orientation. With regards to this notice, we state the following scenarios:

- When the user pushes the device on the left side (so it moves to the right), the accelerometer returns a positive x value.
- When the user pushes the device on the bottom (so it moves away from him), the accelerometer returns a positive y value.
- When the user pushes the towards the sky with an acceleration of A m/s^2, the accelerometer returns A + 9.81 z value (device acceleration minus g-force , -9.81 m/ s^2)

From the application programming perspective, all of the motion sensors event listener functions return multi-dimensional arrays of sensor values. Particularly, an accelerometer event produces gravity force data (float values along with other sensor-specific parameters) for the three coordinate axes, and the gyroscope returns rate of rotation data for the three coordinate axes.

[70] http://developer.android.com/guide/topics/sensors/sensors_motion.html

Next we cover the most important implementation notes, concerning the platform APIs[71] for sensor data monitoring:

- Determine which sensors are available on a device.
- Determine individual sensor's capabilities (maximum swing – 3.5.3, sensitivity – 3.5.4).
- Register and unregister sensor event listeners that monitor sensor changes.
- The sensor data's natural coordinate system never changes, as the device moves.
- The sensor coordinate system is always based on the natural orientation of the device.
- Portrait is not always the default mode. User can hold the device in many ways.
- "Locking" the application to portrait mode solves this issue.
- Regular check current device's orientation and compensate accordingly.
- Register and unregister sensor event listeners that monitor sensor changes.

When a typical smartphone is held normally, the gravity arrow points down, as shown in figure below:

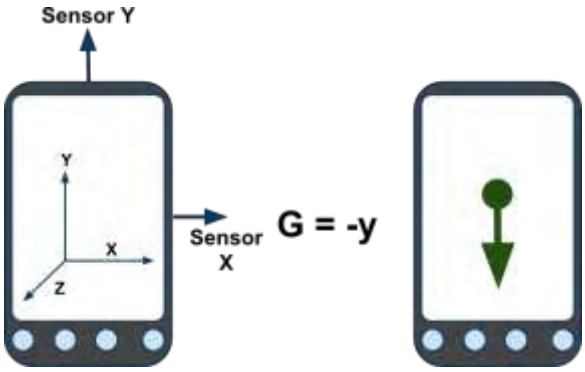


Figure 17: The letter "G" means the direction of gravity force in the sensor coordinate system. G = -y means that gravity is aligned with the device's negative-Y axis, as measured by the accelerometer.

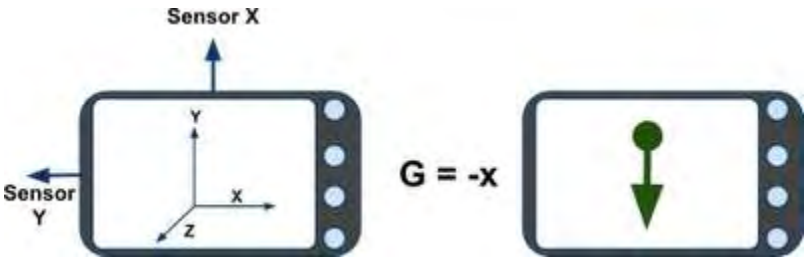


Figure 18: The sensor coordinate system never changes, so in G = -x means that gravity is aligned with the device's negative-X axis.

[71] http://en.wikipedia.org/wiki/Application_programming_interface

3.4 Measuring Distance

Human motion detection and tracking is one of the major challenging applications for an inertial navigation system [6]. There is a wide variety of approaches and implementations depended on the:

- Amount and the precision (quality) of inertial sensors the system is equipped with.
- Mounting point. Whether it is on the body or the shoes.
- Preferred inertial navigation algorithm. Whether they are based on pedestrian dead reckoning (PDR), Zero Velocity Updates (ZVUs), or on both techniques.

3.4.1 The curse of integration

In general, inertial measurements involve differentials of motion and there is no alternative using in order to determine the current position, which means that integration constants are required in order to acquire accurate results. Inertial navigation algorithms involve three phases of sensor data integration cover below.

For one thing, the orientation, or attitude, of a mobile device regarding a local reference frame determined by can be tracked by integrating its angular velocity. The orientation of the device can be specified using several different representations, including Euler angles, quaternions and direction cosines. Rotation vectors though, as described at section 3.2.3 are the most common practice for tracking orientation changes.

Another, integration step of algorithms is the one related to acceleration and initial velocity for estimation of actual velocity. Due to regular update of acceleration data, these values settle out quickly and respond slowly to sudden but short-lived changes in motion. Eventually, the last part of the integration process includes estimating position by integrating the velocity value calculated on the previous step along with an initial position value.

Eventually, due to the fact that the integration steps mentioned above are sequential and rapid occurring (high rate of sensor data), one can easily understand that the overall process is computational demanding and very sensitive in errors occurred.

Next, we provide the whole inertial algorithm process in the following scheme. For detailed notes and analytic math equations we advise referring to section 3.2.1 of reference [11].

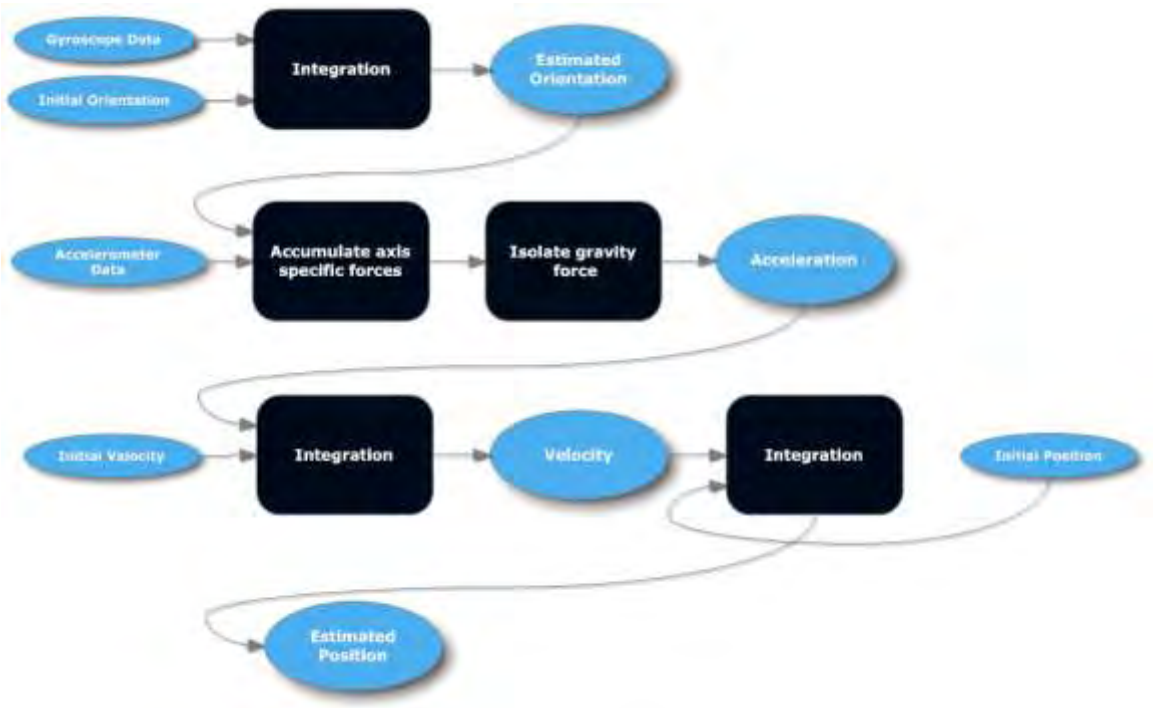


Figure 19: The inertial navigation algorithm [11]

3.4.2 Pedestrian Dead Reckoning

Some pedestrian positioning methods measure steps indirectly in order to determine location [7]. To accomplish that, they suggest that the pedestrian should carry an inertial measurement unit in order to allow the system to perform steps counting whereas other solutions propose that the IMU should be foot mounted in order to achieve better precision. On the other hand, there are some hybrid methods that suggest using insole mounted sensors in assistance with smartphones.

Starting from the latter case, Apple has already engulfed two implementations: Nike+[72] and Adidas micoach[73] pace sensors. More thoroughly, both of these solutions involve an actual sport kit consisting of the proper mounting sport shoes and insole sensor (only one is used for each pair). The concept here is quite simple and actually it includes an Apple smartphone owner equipped with the preceding sport kit. Consequently the user's device can track useful data

[72] http://store.apple.com/us/product/MA368LL/E
[73] http://www.adidas.com/us/micoach/ui/Product/#!/speed_cell

gathered every time the user performs a step. With this dataset, proper algorithms extract intelligent information, such as the number of sprints performed, the total distance ran, maximum user speed and other useful knowledge. Afterwards, a user can synchronize (via Bluetooth, Wi-Fi, or a proper plugin component[74]) all this information acquired with the mobile phone and display it properly.

At this point one may wonder, why the two clothing, footwear and sportswear, supplier giants preferred the above approach while a large and fast increasing number of pedometer[75] applications exists in both Apple's and Google's mobile application market. Is it not the use of inertial sensors itself capable of measuring distance? Apart from the obvious marketing reasons, these two leading industries looked forward to reaching better accuracy. From the early start one can detect, in several degrees, false step detection by using either a free or "well-paid" pedometer application purchased from mobile market. What is worse, when a user is standing still but changes the smartphone's orientation almost all applications perform step counting. During the time this report is written we found in mobile markets pedometers that only use device's accelerometer for step detection. Rolling back to sections 2.1.1 and 2.1.3, we can assure that by using sensor fusion, with accelerometer and gyroscope participation, the estimated results can be greatly benefited, as the latter sensor can provide orientation data. To alleviate the problems traced at the existing "traditional" pedometer approaches, Nike and Adidas resulted in integrating the insole sensors.

In our approach, pedestrian dead reckoning techniques prove inefficient for determining distance based on inertial measurements. In both Nike+ and Adidas micoach pace sensor cases, we highlight the disqualifying features below:

- Purchasing such Sport Kit equipment is opposite to adopting a low-cost solution
- Wearing this gear is against the everyday life gadget needed feature
- Using sensors which battery is not replaceable or rechargeable limits the perspective of developing an autonomous and self-contained system
- False step detection is also imminent (leaning or pivoting to the leg that wears the sensor-mounted shoe, but without changing place)

[74] http://www.adidas.com/us/micoach/ui/Product/#!/connectios
[75] http://en.wikipedia.org/wiki/Pedometer

On the contrary, under pedometers hood lays the principle that they must be calibrated for the stride length of the user and that the user does not perform any running and jogging motions during navigation, but only walk according to normal walking patterns, else they produce large errors.

In either case, digital map aid and supplementary sensors are used to restrain the accumulation of subsequent errors that increase over time due to false step detection. We focus on combined pedestrian dead reckoning with zero velocity updates techniques to reach the desired precision of a mobile phone inertial navigation system.

3.4.3 Zero Velocity Updates

Some human motion detection and tracking methods determine the distance covered using inertial navigation algorithms. Usually, when such algorithms are applied, the position and velocity estimations diverge by some seconds. Things are even worse when estimation error is not resetting. In most cases where a user moves at normal walking patterns, his speed should be close to zero for a short time. Normally, this is the time when his foot touches the ground in a rather periodical way and remains there for a short time (most frequently from 0.1 to 0.3 seconds). These periods, in which velocity regularly becomes zero and the user's foot is stationary (stance phase), are called zero velocity intervals.

Zero velocity updating algorithms aim to detect zero velocity intervals in order to reset velocity error to zero. One effective technique to accomplish that, is observing the magnitude of the angular velocity (gyroscope measurement) and positive detect an interval when it falls below a determined threshold [9]. Positive detection is very crucial for algorithms' performance, but it very tedious, as it greatly dependent on the motion pattern of the person, its velocity, the ground type, the sensor's accuracy and the mounting point of the device. Another feature that plays significant role in the location estimation process is the integration period for accelerations and velocities. Research has proved that most error-free approaches keep the integration interval short (1 – 3 seconds) [10]. One way to achieve these short breaks is mounting the device on user's foot.

3.5 Determining map representation

As stated in section 1.2, a handy mobile user interface should be more of a "smart guide" and less of a "paper map" on a user's smartphone. The application should also involve dynamic data to prove really helpful for the user. From, the given system requirements in section 3.1, it appears that a proper interface design should encompass the three following options:

- Path determination for a given location
- User position indication on map
- "Near me" functionality support

Besides the challenge of implementing a seamless user interface that deals with the above issues in an efficient way, we have to highlight that above all, deciding a suitable map representation can make a huge difference in the performance and path quality[76].

Computer science technology in game programming has made a great leap forward in fields of map representation. There is a certain issue for path finding algorithms that we need to encompass. They are usually worse than linear and when doubling the travel distance, it takes more than twice as long to find the path. In other words, pathfinding is proportional to searching an area of circle shape and when the circle's diameter doubles, the area becomes four times larger. Usually, as the node number included in the map representation decreases, the algorithms tend to be faster.

In the proceeding notes, we state most common representations that are used for "world navigation" in game programming, including their features and properties.

Grids

A grid based map focuses on a dividing a uniform section of the world into small regular shapes, most known as "tiles". In most cases, square, triangular, or hexagonal shaped grid tiles are used. Generally, grid maps are simple in design and implementation, thus they are very common in game world visualization.

When grids are used for way finding, a mobile user is not constrained to move in grids and his movement costs are uniform. A map design can include movement in straight line from one

[76] http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html

tile to another, far away tile when there are no obstacles between the two. Moreover, a user's movement can be simulated as tile, edge or vertices movement.

In games where units move only to the center of a node, tiles are the best solution. Besides the neighboring tiles, diagonal movement may also be permitted assigned with the same or higher cost. On the other hand, in cases where tiles are large (malls or airports), or users are able to move anywhere in the grid coordinate system, edge movement seems to be the proper choice. In this way, a user approaches a tile at one of the edges (usually in the middle) and leaves the tile at the another edge. Eventually, when the world includes obstacles, it maps them by aligning their corners to tile vertices. Therefore, a unit can move from vertex to vertex, as the shortest path around an obstacle will be going around the corners. This approach benefits from the fact that it assumes the least wasted movement, but size of the units is a factor that needs to be predetermined in order to adjust the paths.

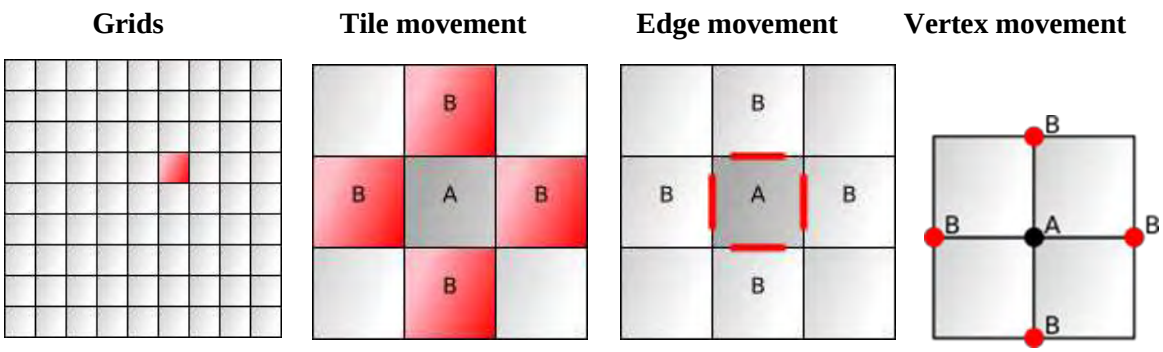


Figure 20: Grid map representations

Polygonal maps

One common alternative to grid map representation is a polygonal one. Non-grid representation are preferred in cased where the units move in straight lines rather than navigating through tiles and their movement across large areas is uniform. Even if some world sections include grid graphs non-grid ones can coexist.

When a graph contains obstacles, the shortest path would be going around their corners. These corners can be considered as elementary navigation points and they have to be redefined on every map change. When obstacles match a tile cluster on a grid, the navigation points will be aligned with the vertices of that cluster. What is more, a map matching algorithm has to know the

predetermined links between navigation points and in the simplest methods a visual map is built by connecting pair of points that can be seen from each other. Another prerequisite for tile maps is that the map does not change its state during navigation, else more sophisticated algorithms will be needed.

When polygonal map graphs contain many open areas or long corridors, connecting every pair of obstacle corners is tedious as if there are N corners (vertices), the graph should have up to N^2 edges.

Polygonal map with obstacles **Worst case scenario for complexity**

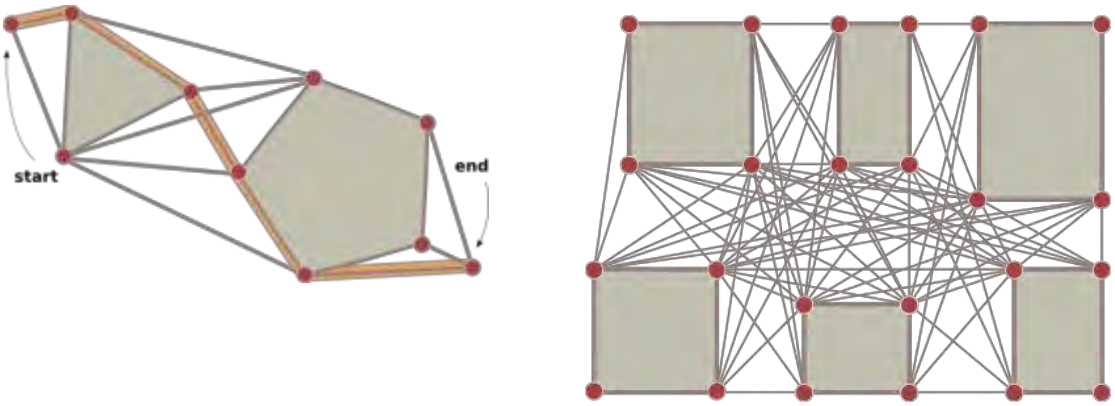


Figure 21: Polygonal map representations

Here's a simple example of one kind of polygonal map representation. In this example, the unit needs to move around two obstacles:

Although there are techniques for removing redundant edges, still the large number of the remaining edges has great impact on battery life and memory usage. The polygonal approach compared to the grid one, speed ups map matching algorithms as the navigation points provide "shortcuts".

Navigation Meshes

Similar to polygonal representation of obstacles, is plotting walkable areas with non-overlapping polygons, a technique known as a navigation mesh. In this method, obstacles do not need to be stored and the walkable areas can exhibit extra information (specific path cost, "requires swimming" etc.).

Navigation mesh movement can be treated in the same way grid movement is treated. In this case, a navigating user can use polygon centers, edges, or vertices as navigation points. When there are no specific restrictions on what part of polygon can be treated as navigation points, there is also a hybrid movement capability that uses both the edge centers and vertices. In a hybrid scheme, polygon centers are rarely useful and along an edge there can be multiple points. Usually, to get around the obstacle, the unit goes through a vertex, but elsewhere, it can go through edge centers.

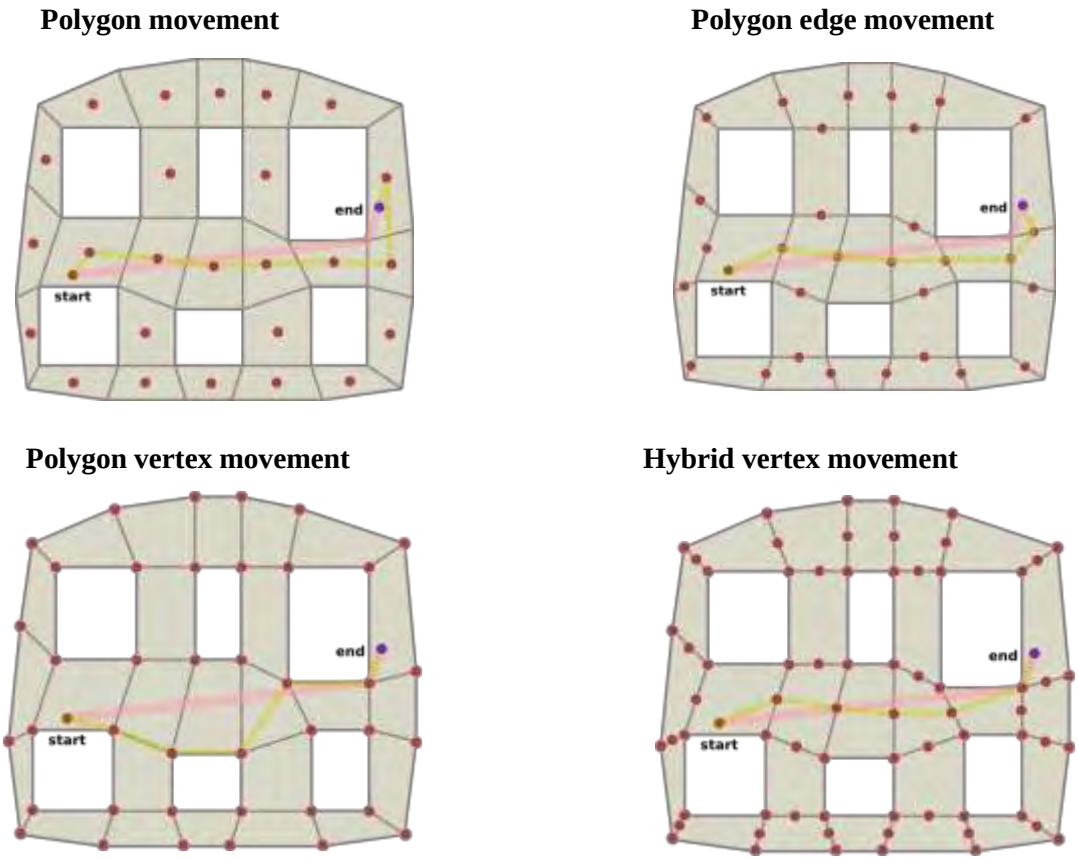


Figure 22: Navigation meshes map representations

Map graph determination

Choosing a proper world representation is not one step process. If an option is not satisfactory, a map designer can always reconsider, and use a different representation.

In many real world scenarios, there are large areas of coverage (multi-floor structures or groups of structures, with multi-room floors) and in these cases creating a simpler graph (navigation

mesh, or hierarchical representation of the grid map) can benefit in great extend map matching algorithms. Generally, grids are more flexible, as they deal with variable movement costs (collision areas, slope, penalties for dangerous areas, etc.) , use very little memory for edges, but use lots of memory for nodes, which has direct impact on algorithms performance.

There are two tools that assist pathfinding and map matching algorithms. One is defining shortcuts by linking edges of non-adjacent vertices. The other involves setting anchor points (waypoints or "beacons") along a path to mark locations that are along good paths. These points can be set manually or computed automatically with a proper algorithm.

Eventually, an ideal implementation of a map design, regardless the opted map graph format from the preceding ones, not only should align with the design principles that we stated at section 3.1, but also encounter the following issues:

- Support for embed social features
- Be in a easily editable and distributable format

For more thorough details on map representations and their impact on pathfinding we advise to look into this Stanford's article.

3.6 Dealing with inaccuracy

3.6.1 Orientation issues

Detecting both device orientation state and orientation change events has a great impact on system's overall performance. A mobile phone inertial navigation application developer cannot assume that the user hold the device in a specific position. Particularly, he cannot take the device portrait mode as a default position, as actually the user is more likely to hold the device in landscape mode, because in this way screens are wider and more proper for navigation purposes. This introduces a few fairly subtle issues that affect greatly the accuracy of inertial measurements carried out, especially the ones coming from accelerometers. Luckily, there are tweaks that encounter these problems, the most popular of them are, rotation vectors, as described at section 3.3.3, which are capable of tracking orientation changes.

3.5.2 Sensor drifting

As described in section 2.4.2 since each new position is subject to last position estimation, an inertial navigation system is a relative system. Every faulty calculation on the first step of the procedure triggers cumulative errors to other computations. In this way, any error occurred at each phase remains constant and all subsequent errors are cumulative, increasing over time. The total number of errors is known as "drift" and can be corrected by calibrating, thus resetting user position based on a known, fixed one (waypoint or visual checkpoint).

In the table below we state the most common MEMS gyroscope drifts and the reasons they appear, issuing the importance of showing deep, purposeful and sincere intent from developers to deal with them.

Error Type	Description	Result of single integration
Bias	A constant bias offset	A linearly growing error in angle
Random noise	Random noise from a variety of sources	An angle random walk whose standard deviation grows proportional to the square root of time
Calibration	Deterministic errors in scale factors, alignments and gyro linearities	Orientation drift proportional to the rate and duration of motion
Temperature Effects	Temperature dependent residual bias	Any residual bias is integrated into the orientation, causing an orientation error which grows linearly with time
Bias instability	Bias fluctuations, usually modeled as a bias random walk	A second-order random walk

Table 4: A summary of the different MEMS gyroscope error sources [11]

The significant error types that occur during gyroscope measurements appear also in MEMS accelerometers. The key difference on accelerometers side is the double integration in order to obtain current position, while gyroscope measurements are only integrated once to determine the

device orientation. In this way, constant bias errors force errors in position estimation that grow proportional to t^2, random noise creates a second order random walk in position and bias instability a third-order random walk.

3.5.3 Bayesian filters

The imminent drift that occurs in raw data obtained from sensors, along with the need for sensor fusion, has assisted in the widespread use of Bayesian filters in positioning systems. Bayesian filters are used in dynamic systems in order to provide probabilistic state estimation on noisy measurements. For instance, in pedestrian dead reckoning approaches, the pedestrian can be treated as a dynamic system (sort of) and its position, with all factors considered such as velocity or angular displacement, stands for the changing state. The best practice for these filters is deploying after the sensor fusion algorithms, combining measurements gathered from all related sensors to form an estimate of the user's position.

In probability theory, a Bayesian filter represents the state of a dynamic system at a given time as a probability distribution over the space of all possible states of the system. This model allows a Bayesian filter to exhibit an estimate of the true state of the system along with a level of uncertainty for that estimate. At this point we introduce the random variable[77] "Belief", which expresses the probability that a state at time t is the true state of the system, combining all of the measurements carried out until that time.

The major issue here is that the complexity of computing Belief at a current time increases exponentially when the number of produced measurements increases. In order to suppress this vast increasing computational requirement, Bayesian filters assume that the dynamic system is a Markov process[78]. At this model, the next state depends only on the state at time t and all measurements made at that time are quantity-variant only to that state at that time. Systems that adopt Markov process are defined by two distributions, a propagation model that defines how the system's state changes over time and a measurement model which defines the probability of making an estimation given the current state.

There are some applications where we have to update the current state despite the fact that there are no new inertial measurements. In this case, the prior distribution is defined and is adapted as the new posterior one. We can also use the precalculated posterior distribution as the prior in

[77] http://en.wikipedia.org/wiki/Random_variable
[78] http://en.wikipedia.org/wiki/Markov_process

successive iterations during the correction step, in order to apply multiple measurements in a single update.

Consequently, Bayesian filters enhance sensor fusion, combining measurements gathered from all related sensors to determine an estimate of the user's position. Although, these filters provide a framework for probabilistic state estimation on noisy measurements, their definition does not specify how such a filter should be implemented. There are various implementations types with varying computational requirements and development costs, the most common of them are described below.

Filter name	**Belief**	**Propagation & measurement models**	**Computational requirements**	**Implementation difficulty**	**Comments**
Kalman	Gaussian	Linear	Low	Low	-
Linearized Kalman	Gaussian	Linear about approximate trajectory			Approximate state trajectory must be known in advance.
Extended Kalman	Gaussian	Non-linear			May diverge in unusual situations.
Multi-hypothesis	Mixture of Gaussians	Non-linear	Medium	High	Heuristics needed to decide when to add and remove hypotheses.
Grid	Arbitrary (discrete)	Arbitrary (discrete)	Very high	Medium	State space must be finite and discrete. Particle filters are generally better unless state space is naturally discrete.
Particle	Arbitrary	Arbitrary	High	Low	Can dynamically vary the number of particles as needed.

Table 5: Summary of the most used Bayesian filter implementations

When deciding which a filter to choose, a developer should balance the trade off the computational requirements with the restrictions that emerge when selecting the proper Belief distribution, the propagation and measurement models. When Belief's distribution is unimodal[79], the best practice is to prefer regular and extended Kalman filters. However, there are cases when Belief's distribution is more complex and multimodal[80], where computationally expensive filters should be applied such as particle or grid-based.

For further notice on Bayesian Filters that are used in indoor navigation systems we strongly we advise to look into sections 2.3 and 2.4 of reference [11].

3.5.3 Sensor hardware limitations

As already mentioned, human motion sensing is a challenge for inertial navigation systems. In mobile phone context though still, we encounter some exclusive hardware limitations of inertial sensors, due to space reserving and manufacturing cost suppression of the devices. There are two native hardware sensor features that affect motion detection process performance.

The first, regarding to accelerometer sensors only, is called "Maximum swing"[81] and refers to the maximum gravity force (see section 2.2.2) that the sensor can detect. In cases of measuring tilt, or shake gestures using earth's gravity, a ±1.5g accelerometer will be more than enough. For vehicle (car, airplane or robot) motion measurement ±2g detection should be satisfactory. In projects where the mounting device performs really sudden starts or stops, a ±5g or more may be needed. Today, Android and iOS mobile devices have ±1 g maximum swinging MEMS accelerometers.

The second hardware feature that applies in both gyroscopes and accelerometers is sensor sensitivity. Sensitivity, in particular, corresponds to the sensor event "listening" update interval. In general, the more sensitivity or in other words the minor the interval, the better in terms of accurate readings. This means that for a given change in acceleration, there will be a larger change in signal (refer to next section for more details).

Our research in the field has not directly issued that the preceded hardware limitations are a problem for performing smartphone owners' indoor navigation. Particularly, concurrent works

[79] http://en.wikipedia.org/wiki/Unimodality
[80] http://en.wikipedia.org/wiki/Bimodal_distribution
[81] http://www.dimensionengineering.com/info/accelerometers

and implementations do not even mention these parameters, but in our opinion they should be encompassed in any implementation.

3.5.4 Choosing an Appropriate Update Interval

Back into modern smartphones inertial sensors, in terms of battery reserving and application overall performance, it is very crucial to choose an interval that minimizes the number of delivered events when deciding the update interval for measurement events. In practice, few applications need an interval update of 100 times a second. Using a lower frequency improves battery life, while a high rate implementation is necessary for high-frequency motion detection, which is the case we focus on this report. The following lists some typical update frequencies depending on usage scenarios. It is obvious that a high-frequency update interval has a great impact in both sensor drifting and computational requirements of sensor fusion determining the proper Bayesian filter implementation in large extent.

Event frequency (Hz)	Usage
10–20	Suitable for use in determining the vector representing the current orientation of the device.
30–60	Suitable for games and other applications that use the accelerometers for real-time user input.
70–100	Suitable for applications that need to detect high-frequency motion. For example, you might use this interval to detect the user hitting the device or shaking it very quickly.

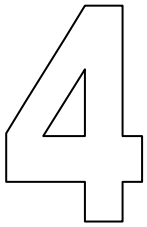
Table 6: Common update intervals for acceleration events[82]

82 http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/MotionEvents/MotionEvents.html

3.5.5 Constant Velocity vs Zero Velocity

In both Pedestrian Dead Reckoning and Zero Velocity Updates approaches, detecting the state when a user actually stops moving is very crucial for the overall performance of the applied algorithms. Yet, another key factor is discriminating constant velocity against zero velocity. In physics, constant velocity imposes zero acceleration and when double integrating the later, the distance covered should be indicated as zero too, during accelerometer measurements. Such false estimations have great impact on system's consistency and dealing with constant velocity is a rather tedious process. Throughout our research, we have not yet traced a deep, purposeful and sincere intent for encompassing this issue and reaching a global (development case insensitive) and robust solution, except counting the total zero velocity values that appear during a rather abstract interval.

4. Existing Implementations – Evaluations



This chapter includes four system implementations and evaluation analysis of them. Each of these systems stands for a different approach of indoor positioning with smartphones. The first is the outcome of the Bachelor thesis, submitted to Worcester Polytechnic Institute by three students, and it represents a hybrid model based on Wi-Fi signals and inertial measurements to determine location. The second and the third are the products of indoor positioning service providers, Qubulus[83] and Wifarer[84], which rely only on Wi-Fi "signal scans". The last, incomplete though, is the sum of our efforts to develop a system based exclusively on inertial sensors.

4.1 Bachelor thesis at Worcester Polytechnic Institute

This is the effort of three students, Manh Hung V. Le, Dimitris Saragas and Nathan Webb, submitted on October 22, 2009 to Worcester Polytechnic Institute, Massachusetts, USA for partial fulfillment of the requirements of their Degree of Bachelor of Science.

The objective of this thesis was to design and implement an indoor navigation system on a web-enabled smartphone. Particularly, measurements from device's wireless adapter where collected with location fingerprinting technique described at section 2.4.1 while accelerometer and compass data were used to repeatedly approximate the device's motion. Dijkstra's[85] algorithm was chosen for its low computational complexity, for calculating the optimal path from user position to destination.

The map representation model was one in which the rooms and hallways of the building were visualized as graphical nodes and branches. In this approach a map consists of four different elements: rooms, hallways, stairs/elevators, and large rooms and all of these elements are represented by nodes with coordinates and information associated with them. A link is a line on which a user can traverse the distance between two nodes. The user can move from a node to a different node only when there is a link between them.

[83] http://www.qubulus.com/
[84] http://www.wifarer.com/
[85] http://en.wikipedia.org/wiki/Dijkstra's_algorithm

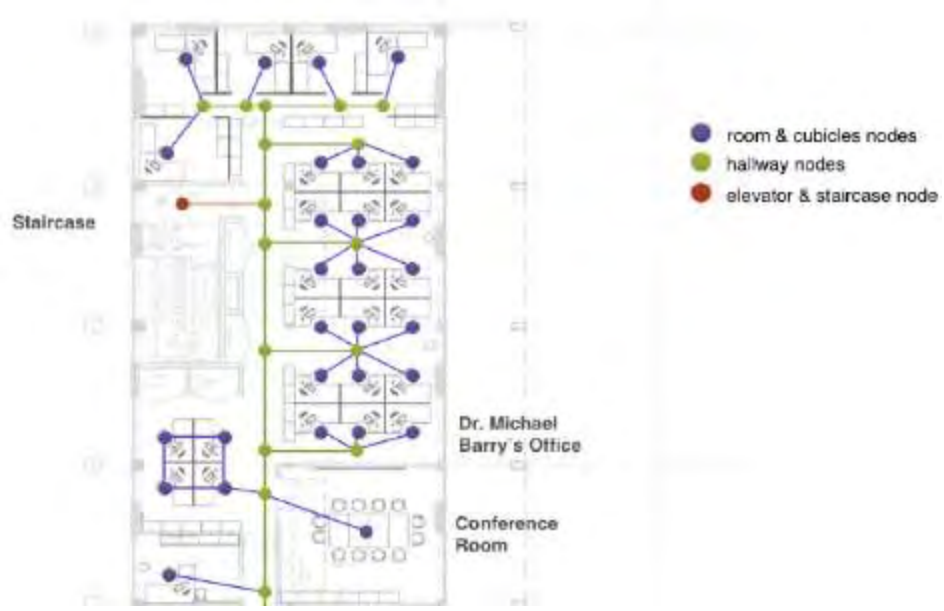


Figure 23: Example of map graph used [13]

The design and implementation device that was used is the HTC Hero, a smartphone with Google Android platform[86] deployed. Below, we provide some screenshots of the developed application.

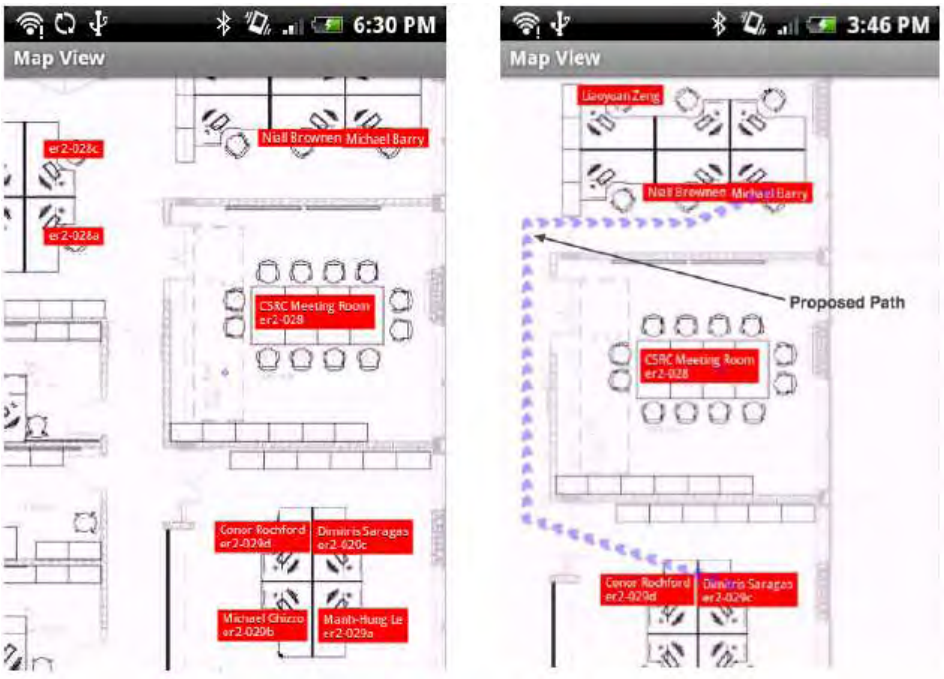


Figure 24: Map View screen with and without route directions [13]

[86] http://www.htc.com/ca/support/hero-telus/tech-specs/

The Wi-Fi position results of the implemented system in sample test scenario are given below.

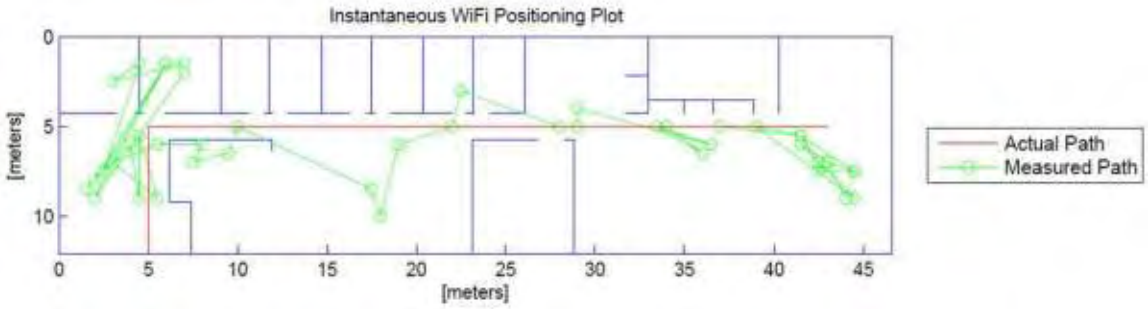


Figure 25: Instantaneous Wi-Fi Positioning Plot [13]

From a more detailed view to above graph, one can see that positioning indicator is randomly moving. This is more obvious at the dense "zigzag" path lines. Following this notice, another thing is that the positioning indicator went through walls or even jumps to the next room and back. However, there are two optimistic observations. For the first thing, regardless the actual position in the building floor, randomly or not, the positioning indicator never goes outside the outer walls. Moreover, some spots have very good hits in terms of device's actual position. These are the green circle spots that match the red line of actual path.

The positioning system implemented on the HTC Hero makes use of inertial measurements along with Wi-Fi positioning to keep a running estimate of the user's position. Enhancing Wi-Fi measurements plotted in Figure 25 with motion and position sensor data results in the positioning estimates plotted in the figure below. A comparison of these two figures makes clear the improvement.

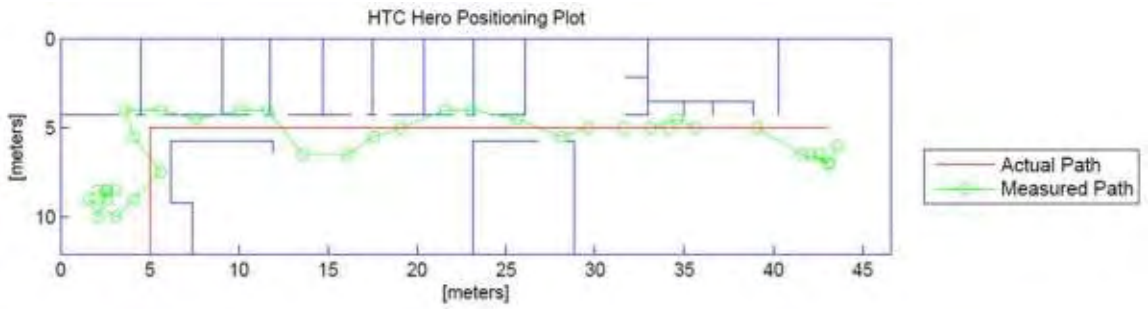


Figure 26: HTC Position Estimates [13]

To further improve the position estimate, each point is mapped onto one of the links shown in the graph below. These links connect the nodes that define locations and intersections in the

application's routing algorithm. Each link endpoint that is not an intersection is a room, a cubicle, a desk, or a stairway.

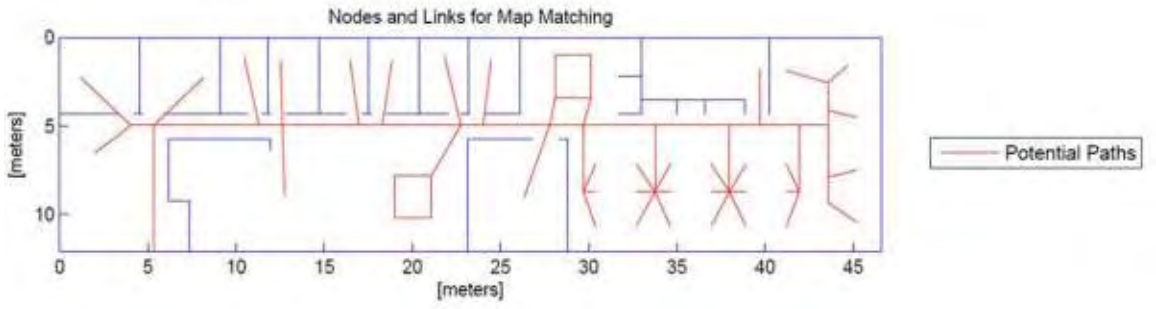


Figure 27: Nodes and Links for Map Matching [13]

Improved performance after map matching is evident in this plot, where nearly all of the points map to the user's actual path. The average distance between the estimated positions and the user's path is now 0.71 meters. This accuracy is sufficient for indoor navigation.

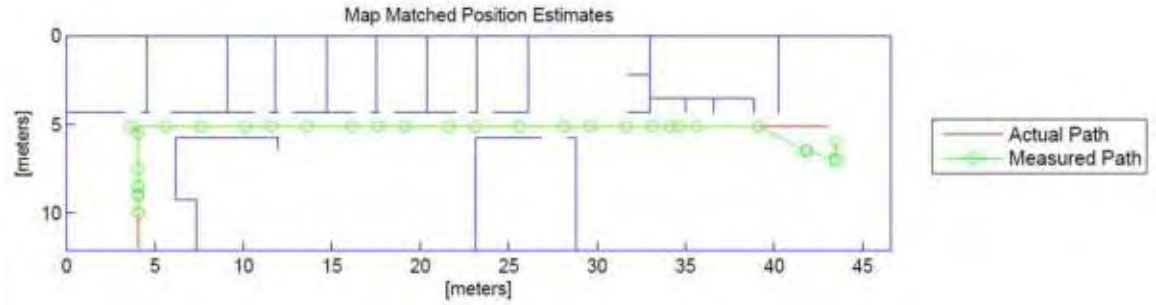


Figure 28: Map Matched Position Estimates [13]

To sum up, the thesis implementation described above verified that two meter accuracy, sufficient for navigation, was achieved, which shows promise for future handheld indoor navigation systems that can be used in malls, museums, hospitals, and college campuses.

4.2 Qubulus Cloud Positioning Service

As stated above, Qubulus[87] is a cloud[88] positioning service provider of positioning platforms for smartphones, where GPS does not work. In addition, Qubulus provides a cloud positioning API for Android and we have to admit that it is the first in its category to make a useable, indoor positioning API available to individual and enterprise developers alike.

[87] http://www.qubulus.com/about/
[88] http://en.wikipedia.org/wiki/Cloud_computing

Figure 29: Visualization of Cloud Positioning API information flow[89]

Qubulus's solution to indoor navigation proposes a Wi-Fi fingerprinting method as described thoroughly in section 2.4.1. Once fingerprint data is collected, the Qubulus Positioning Engine can start positioning the device. Using the devices own knowledge of multiple radio access technologies, the device will report the details about its radio environment to the nearest Qubulus Cloud Positioning Server to match and estimate the user's current position based on the existing fingerprint dataset. Particularly, the process of developing an Android application for a specific site involves the following steps:

- Digital map creation (graph from actual floor plans)
- Recording data, by walking around the site with the device (Wi-Fi fingerprint dataset)
- Export data and send them to Qubulus, to get back filtered data of extracted collection
- Deployment (installing the application in the device and running it)

During the time we build this report, Qubulus Position API was at beta version and free of charge so we had the opportunity to create a sample site for evaluation. Unfortunately, the API now has turned to commercial and Qubulus charges the support.

For the trial testing we used again an HTC Hero device with Android version 2.2.1 , Wi-Fi enabled and 2G/ GSM mode only as the API proposed, for better accuracy of fingerprint collection. Next, we chose a house as a test-bed and the main reasons for that were "satisfying"

[89] http://www.qubulus.com/products/positioning-api/

WiFi presence (4 overlapping Wi-Fi signals per room) and availability of exact floor plans. We made an initial map in .png format of a real floor plan for a single floor, investing too much time, trying to be as accurate as we could, as we thought it is maybe the most crucial initial configuration feature (as well as Wi-Fi presence).

Afterwards, we placed the digital map on Google Earth to get the required coordinates, as Qubulus advised. Here too, we tried to be as accurate as we could, but Google's Earth resolution was not too good at the territory. Moreover, we carried out the recording process, trying to be very strict at this, by making the floor as "crowded" with measurement spots as we could and following the API's manual notes to the best. We took measurements from exactly 13 spots, trying to make them not overlapping, but as close to each other we could.

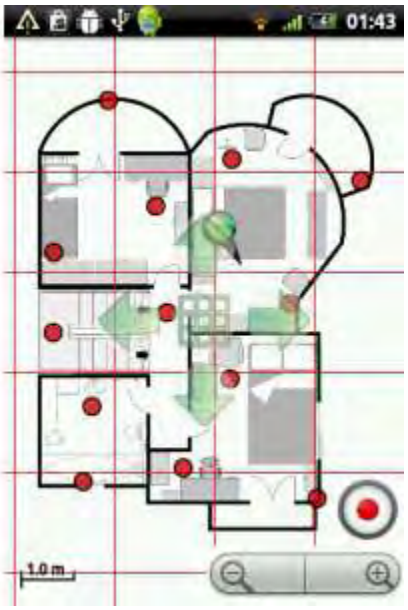


Figure 30: Digital map corresponding to exact floor plans

Figure 31: Recording spots. Red spots mark the Wi-Fi measurement collected data

The later step was, to send the fingerprint collection to Qubulus via email and get back the processed data, in order to proceed with the deployment. The following figure states the testing results.

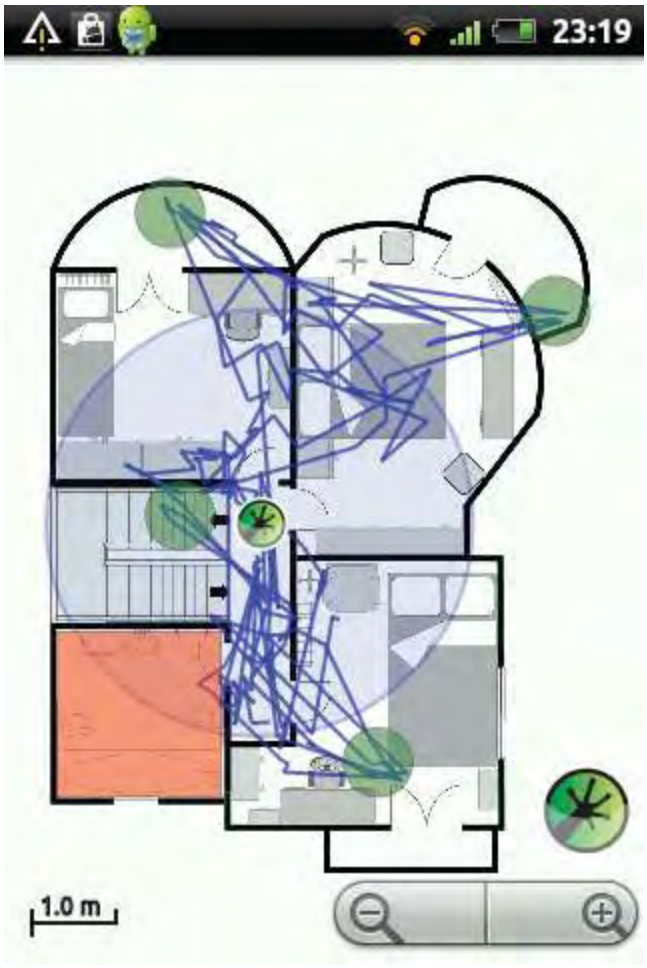


Figure 32: Path history of a localization test results

Aligning with the observations of the thesis described at the previous section, when inertial measurements are not available and Wi-Fi data from signal scanning are only used, one can see that positioning indicator is, again, randomly moving. This is more obvious at the dense "zigzag" path lines. Following this notice, another thing is that the positioning indicator went through walls or even jumps to the next room and back. A first noted and weird case was that there was one room (the red transparent rectangle highlighted one) where the Position Engine never positioned the device there, regardless if the correct position was inside the room or not and whatever the experiments we carried out. We guess it was a "badly recorded spot" or a spot with bad Wi-Fi presence.

However, as in the Bachelor thesis, there are two optimistic observations. For the first thing, regardless the actual position in the building floor, randomly or not, the positioning indicator

never goes outside the outer walls or balconies. Moreover, some spots have very good hits in terms of device's actual position. These are the green transparent circle spots that match the red line of actual path.

Eventually, we posted a support request to Qubulus Support and Management team stating our efforts and our beliefs for the observed testing results, assuming that better accuracy would be achieved with a more exact building map, best fitting coordinates in Google Earth, better Wi-Fi presence, more thorough recording process with even a new generation Android Device with better Wireless and Mobile Antennas.

They replied back, with an information note where, in quick words, they advised to change to site resolution, thus picking up a larger building or area for localization and changing the density of the wireless access point coverage. They proposed a model in which none access point signals overlap and there is at least one access point per room.

Putting all together, experimenting with Qubulus Positioning Engine and taking the previously mentioned thesis conclusions into consideration, we strengthen our initial belief that, in terms of obtaining a more accurate estimation of potion, it is necessary to elaborate a new approach that relies mostly on inertial measurements and less in Wi-Fi "signal scans".

4.3 Wifarer Indoor Position Technology

Like Qubulus, Wifarer[90] is another indoor navigation service provider, which promises to accurately guide visitors around a venue in real time, combining the venue's Wi-Fi system with the visitors' smartphones. From the early start, even if the current Android application is free, one can see that Wifarer is rather commercial-oriented than a general purpose indoor navigation application.

Unlike the previous mentioned provider, Wifarer seems a little more immature or at least not evolving in the same rate Qubulus does for now, as it is currently in beta version at a few selected locations in the US and Canada, including airports, universities, convention and shopping centers. However, it promises to soon deploy on iOS installed smartphones (current is available in US and Canada Android Play Store), including more site and venues around countries.

[90] http://www.wifarer.com/

Figure 33: Wifarer Position Engine[91]

Due to our location unavailability (outside US and Canada) we could not test or deploy the application, thus we will not provide any review here, but leave the call to the readers of this report to qualify this candidate. In that direction, one can watch a demo video[92] posted at the company's site.

4.4 Our efforts

At the early start, about two years ago and back in the third quarter of 2010, when we began to orientate in the field of inertial navigation, gyroscope sensors were not deployed at any smartphone (section 2.3), but there were just rumors about it. Immediately, we came across with a significant potential opportunity. To achieve indoor navigation on a modern mobile phone based exclusively on gyroscopic and acceleration sensor data. Unfortunately, all the way round, the harsh truth was that this proved unfeasible, especially in regards to the time horizon of a Master Thesis fulfillment and our limited resources, human and financial ones. Our only asset, in terms of equipment, was an iPod Touch 4rth generation[93], with both gyroscope and accelerometer sensors installed, and that was our conscious choice, along with a major challenge and motivation too.

However, letting aside all the pessimistic indications during our efforts, we remained dedicated and stuck to the initial plan. Not to use any Wi-Fi sensor for assistance, as we were convinced

[91] http://www.wifarer.com/benefits/capabilities
[92] http://www.wifarer.com/users/demo
[93] http://www.apple.com/ipodtouch/specs.html

that Wi-Fi tracking methods (section 2.4.1) do not have sufficient resolution for implementing an indoor navigation on mobile phones, as even nowadays there is no traditional industry giant company in this direction. In addition, adopting such techniques was against our desired designing features of the developed system, described at section 1.3 of this report.

For one thing, our first application that was installed on iPod Touch was a rough pedometer application in a pedestrian dead reckoning approach (section 3.4.2), without any additional sensor, expect device's accelerometer. Before our first implementation steps, we conducted a survey on Apple's App Store for popular pedometer applications, free and paid ones. Their purpose was to display the user's distance covered during a walk or run on outdoors. Some of them took the process a little further from step counting in order to measure distance, by requesting user input of height, gender and weight, to reach a more accurate stride estimate. Nonetheless, actually all tested applications involved a major omission. They did not mind the device's orientation, thus producing many false positive step detections on device's orientation state change, even in cases the users, holding the phone, were stationery.

In that direction, we implemented an abstract pedometer application, with gyroscopic data displayed on main screen for trial testing and battery usage observation reasons only, when the two sensors were combined. Gyroscope measurements never involved at any phase of the step counting procedure.

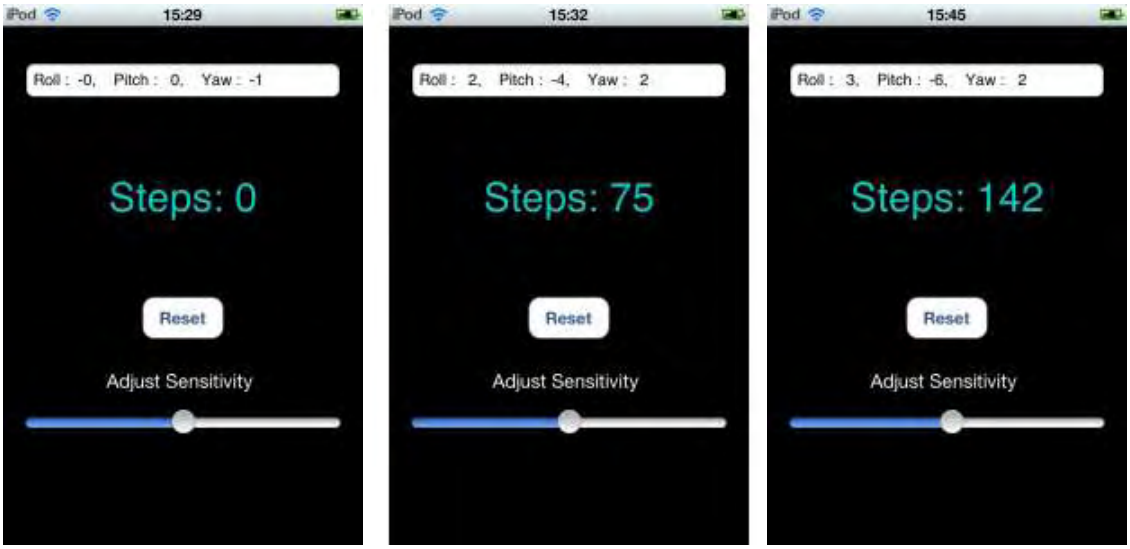


Figure 34: Screenshots of pedometer application

Minding this approach, soon enough we realized that a pedometer, based on accelerometer alone, is a lost case when attempting to deploy an inertial navigation application in a smartphone. Even when we adopted typical motion patterns (advised from [10]) for walking and variable thresholds for acceleration and velocity, the produced results were near the ones of the existing pedometer applications in App Store, as mentioned above. The last attempt was to add a scroll bar to the application's user interface with which we could adjust the pedometer's sensitivity. The left side of the scrollbar, with less sensitivity, produced fewer steps than the actual performed, while the right side, with higher sensitivity acceleration thresholds for step counting, produced many false positive step detections.

Eventually, regardless the pedometer implementation approach, we were convinced that this design direction is not efficient due to the following reasons: device orientation and user's holding stance are not encompassed. Side or backward stepping could not be detected, (no existing application subtracts steps) and this alone disqualifies pedometers from being a valid candidate for indoor navigation.

Afterwards, as expected, we turned our hopes to zero-velocity updates approaches, relying on implementation principles of an enhanced algorithm for zero-velocity intervals detection as described in [10]. The concept here was to define a predetermined minimum number of subsequent accelerations and when this number was collected to calculate the norm of minimum average of the acceleration in the interval. If that fell below specific a threshold, corresponding to the user's motion pattern (crawling, walking or running), a zero velocity interval was detected and error was reset.

Moreover, here again we added a scroll bar to the application's user interface with which we could adjust the thresholds for acceleration and velocity. The left side of the scrollbar, with lower sensitivity, detected less zero-velocity intervals than the actual performed, while the right side, with higher acceleration thresholds, produced many false positive zero-velocity detections. Again, on this implementation, gyroscopic data displayed on main screen for trial testing and battery usage observation reasons only, when the gyroscope was combined with accelerometer sensor. Gyroscope measurements never involved at any step of zero-velocity detection procedure. In addition, like at the pedometer application, device orientation and user's holding stance were not encompassed. Side or backward stepping could not be detected, while acceleration

measurements were collected towards the Z-axis of the iPod's coordinate system. Particularly, during the testing, we walked holding the device at the default position, in portrait mode, with the screen facing our chest, as it would be if we carried it inside a shirt's chest pocket. The testing results varied, depending on the picked algorithm's thresholds and parameters, from some chaotic success stories (measuring a corridor's actual length by walking through edges with precision of 2 meters), to total failures (indicating the corridor's length ten times larger). This effort proved to be much inconsistent, far from reaching a general purpose solution to the problem.

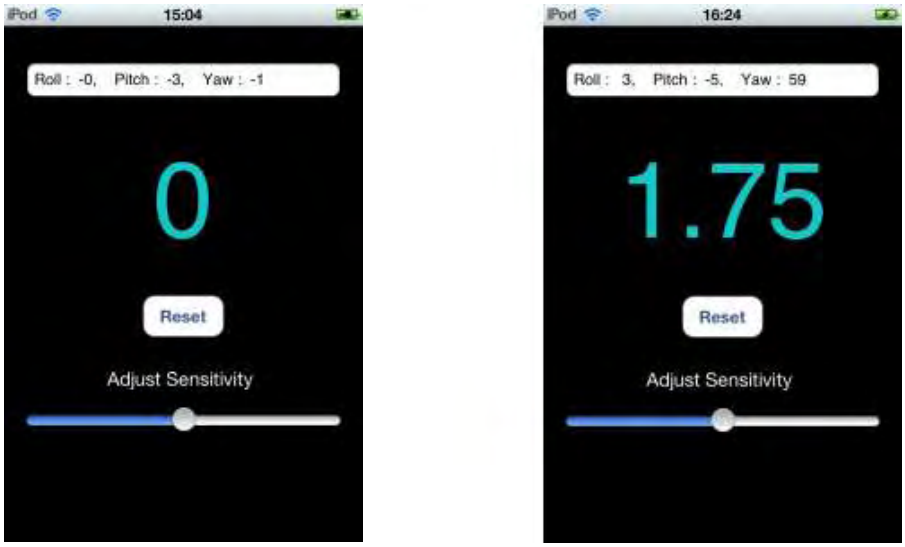


Figure 35: Zero-Velocity Updates approach, displaying distance walked in meters. No direction change detection

A similar attempt to the last, was to detect a direction change to the same axis of the movement (walking towards, then stop and go backwards) and estimate its impact on the distance covered. The results here were totally misleading. The thing about deacceleration (regular decrease in acceleration) is that it appears to be a whole new field in terms of calculating distance from acceleration's double integration. For example, it is quite valid to claim that a speed of 0.8 m/s is twice the speed of 0.4 m/s. But, accelerating an object to reach velocity of 0.8 m/s takes more than twice the energy required to accelerate the same object to get a 0.4 m/s velocity [14]. As mentioned at section 2.2.2, an accelerometer measures weight per unit of mass, a quantity with dimensions of acceleration that is sometimes known as specific force, or g-force (although it is not a force). Those two statements highlight the complexity of calculating the effects of motion's direction change in distance estimation procedure. In our case, the tests showed great inconsistency as there were some chaotic success stories (measured distance of 1,5 meter when walked 2 meters forward and 0,80 meters backward), to total failures (indicating the measured distance six times larger or smaller).

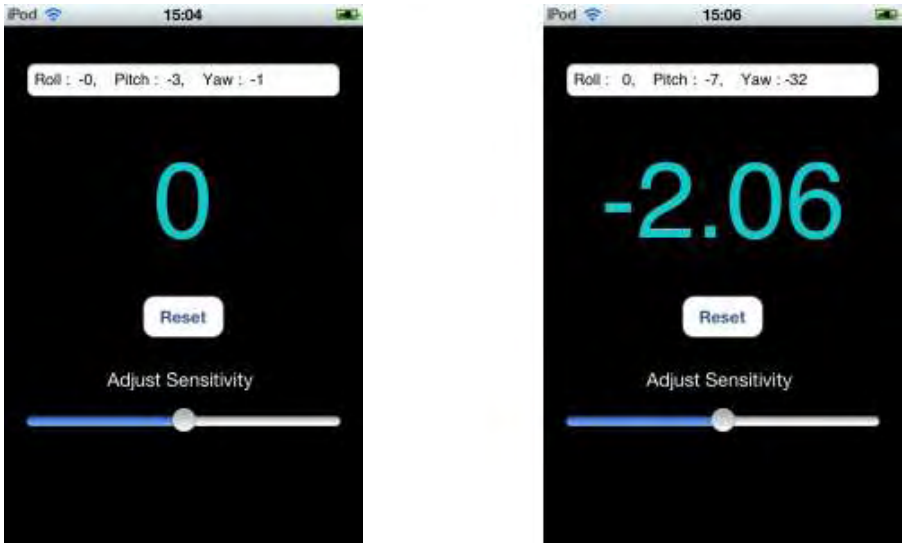


Figure 36: Zero-Velocity Updates approach, displaying distance walked(meters), with direction change detection

At this point, we decided that we had to roll back, in developing a basic scenario of tracking the smartphone's acceleration, thus velocity, when it moved in a way that simulated a constant velocity motion towards on axis. In that direction, we mounted the iPod at a toy car, using springs as gimbals (section 2.2.1) and our attempt was to simulate a constant velocity movement of the car (and the device accordingly), by pulling the power cable which was plugged in at the built in dock connector[94] of the device. We preferred springs as gimbals as we thought that they present higher absorption of iPod's vibrations as we pulled the toy car, thus limiting the impact on measured acceleration.

Figure 37: Mounted iPod on toy car, for tracking acceleration and velocity on Y axis

[94] http://en.wikipedia.org/wiki/Dock_connector

Our motivation for this effort was some findings which noted the differences between acceleration and uniform motion velocities. “Particularly, in the strictest scientific sense, there can only be one kind of velocity and it is the relative speed between two objects in different inertial states. In such case, either of two such objects can be considered stationary relative to which the other object is in uniform motion. Acceleration then, is really a change in such motion. Whereas in the first case we are defining the relative velocity between two objects, in the second case we are transitioning from one inertial state to another. The importance of this distinction between uniform motion and acceleration involves, among other things, the distances traveled during an interval of time” [14].

In order to decompose the above statement, we conducted three test scenarios concerning the iPod mounted at the toy car. Both test versions, included functionality for recording, saving the collected accelerometer data and resetting the collection by erasing the records. The three scenarios involved pulling the toy car at medium or constant high speed and variable speed.

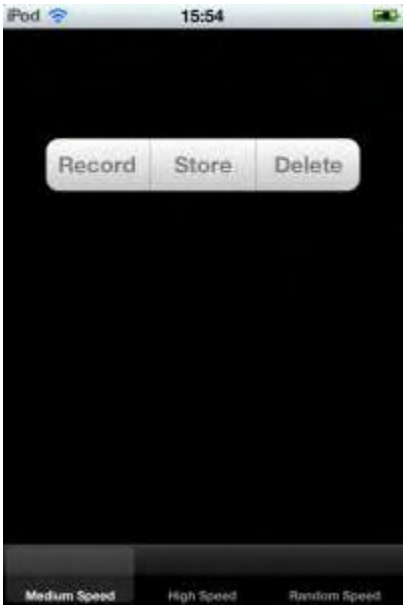


Figure 38: Test application for tracking acceleration and velocity when the toy car was pulled

Our records were stored locally to the device’s memory and we exploited them, by transferring on a PC and plotting them on graphs. Medium and high constant velocity test cases showed significant similarities on the results, thus we cover only the first. Pulling the device at variable speed on the other hand, produced rather uninterpretable results and for this reason we omit to comment the generated graph.

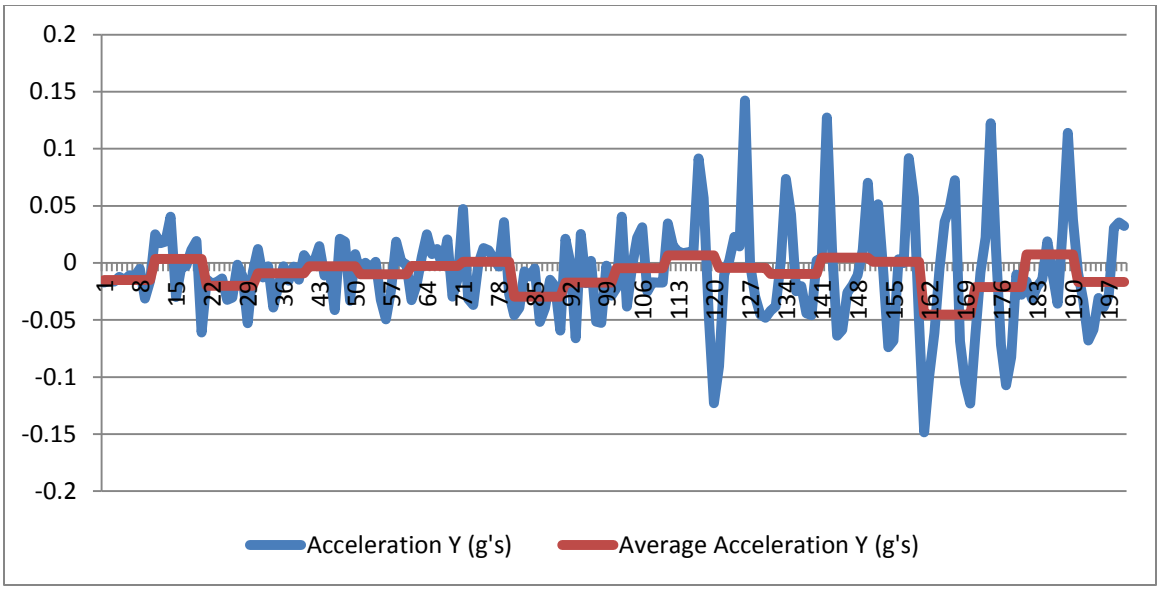


Figure 39: Acceleration data at variable velocity, with direction change. X axis is time in deciseconds

At all of the three cases mentioned above, an accelerometer update interval of 0.1 seconds or 1 decisecond[95] was chosen and the motion of the pulled toy car, with iPod embedded, was two directional. This included a forward motion and the same motion in the opposite direction, back to the starting point.

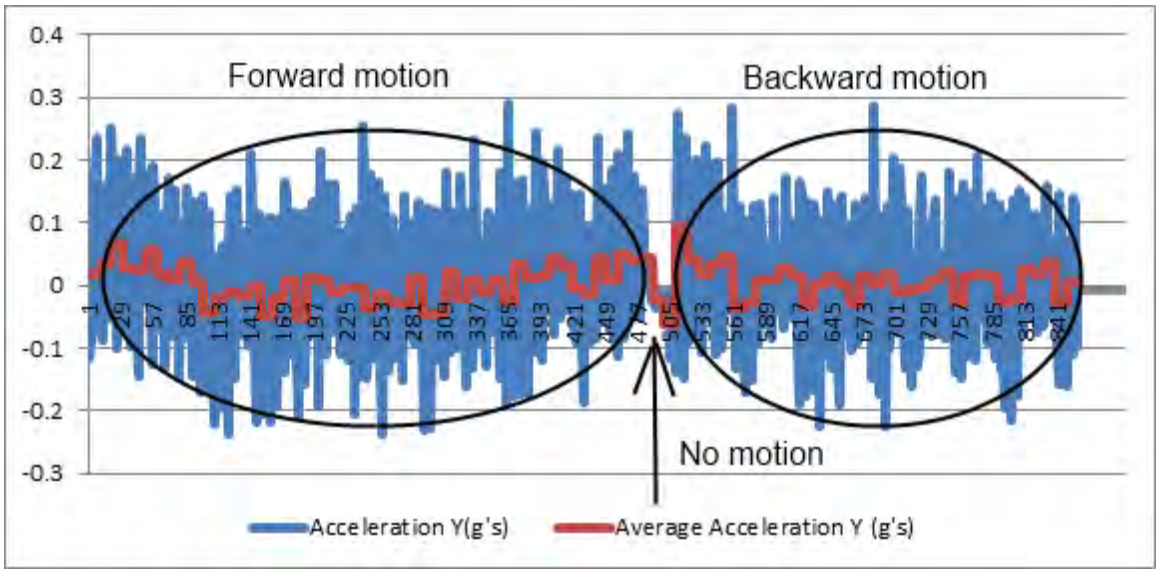


Figure 40: Acceleration data at high constant velocity, with direction change. X axis is time in deciseconds

[95] http://en.wiktionary.org/wiki/decisecond

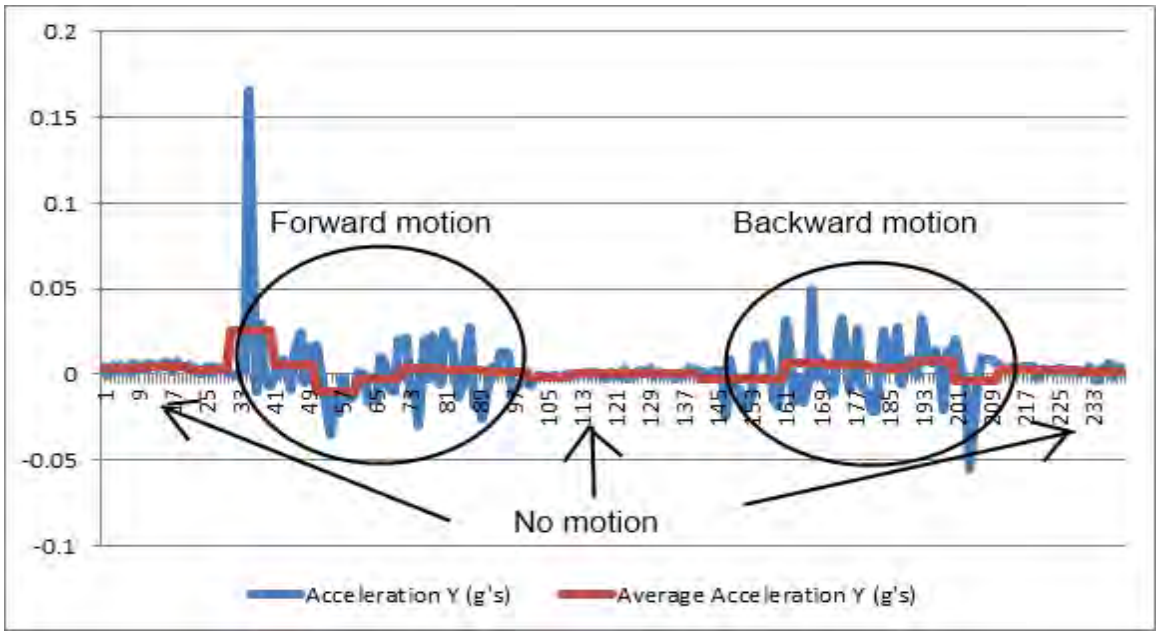


Figure 41: Acceleration data at middle constant velocity, with direction change. X axis is time in deciseconds

Every second, thus every 10 deciseconds, we calculate the average acceleration of that interval to help our investigation of the collected data. By observing the two above graphs, one could fathom that by double integrating the acceleration data through the whole recording interval, the indicated distance covered should be close to zero, as displacement is a vector quantity and the car moves towards and back to the starting point. In this way, the positive areas that are formed under the line of average acceleration intervals and above X axis, should be of the same absolute value of those areas formed above the line of average acceleration intervals and below X axis, when summing up.

However, this is not the exact case for both figures 40 and 41. Moreover, it appears that some peaks or troughs[96] in the graph presented when a significantly larger than the average acceleration was performed and that fact alone messed the correct calculation of total displacement. This strengthens the belief that distinction between uniform motion and acceleration should be always in consideration, when determining the distances traveled during an interval of time [14].

Another interesting finding is that the sub-graphs of forward and backward motion were somehow reversed or horizontally flipped. This has a meaning, considering the fact that

[96] http://en.wikipedia.org/wiki/Trough

acceleration too is a vector quantity. However, the sub-graphs were not identically reversed and this again raises questions of deacceleration's impact on measuring distance.

Eventually, by taking a closer look at the presented graphs, one can merely see a slight off, indicating a hardware bias of the accelerometer sensor. This can be corrected by, "thrusting" X axis a little higher so that summing positive and negative rectangle areas plotted, by average acceleration and X axis, results to an almost zero value.

Consequently, we took the courage to write down our efforts with as much detail as possible in order to help the readers of this report to become more familiar with the current framework of exploiting smartphone's inertial sensors. Moreover, we commented out the produced results with great reluctance, putting a lot of care when describing our implementation steps, in terms of preventing the readers from becoming disorientated in the field.

5. Conclusions - Future Research

Throughout this report overview, a reader can see at first glance the deep theoretical background and the variety of the implementations in the field. Wi-Fi or inertial based, current indoor navigation techniques appear either too deployment case-sensitive, “expensive” due to initial configuration facilities, or inefficient. In addition, there is not a single, consistent and general purpose theoretical framework to follow and sparse information makes it really difficult for a newcomer in the field to start with.

For one thing, Wi-Fi tracking methods (section 2.4.1) for position determination, comparing to inertial ones, seem more intuitive for design and implementation, as theory is well covered and hardware technologies are rather fixed and easily adopted. Radio signal scanning techniques, however, present significant initial configuration cost, in terms of equipment and facilities, with solutions varying from GPS repeaters, ultrasonic and radar transmitters to wireless access points, bluetooth and infrared beacons.

In the world of inertial sensors, things are even more chaotic. There is a huge amount of sensors and implementations for one to mingle with. From magnetic field, motion and position sensors to foot, arm, wrist or even pedestrian pocket and arm mounted approaches, combined with Wi-Fi tracking algorithms and map matching techniques with various map representations (section 3.5).

No matter the amount and the implementation variety of concurrent works in the field, until nowadays the traditional challenge of developing an inertial navigation system still poses an unclaimed trophy. Soon enough, we focused our efforts to deliver a consistent and as detailed as possible, review of existing implementations and provide our speculations it that direction.

Future research

Looking towards the industry side, there are some implementation omissions in both inertial and indoor navigation fields. We state the most significant ones, from our point of view below:

- No map matching algorithm collaborates with gyroscopic data
- Z-level positioning (floor elevation for instance) is not treated with care
- Any implementation should work on outdoors too
- Crowd sourced map model are not, in general, adopted

From our part, we won't tend to let down the effort in designing a mobile phone application capable of inertial navigation. Nonetheless, our first attempt will be to further investigate the current Wi-Fi fingerprinting candidates (section 4.2 and 4.3) and we plan to deploy another test application of Library's and Information Center University of Thessaly[97] structure, as soon as the building gets sufficient Wi-Fi coverage (expected to be done until fourth quarter of 2012).

In addition, in terms of inertial measurements, we will continue our survey for concurrent work in the field and we will end up into acquiring an even more robust and deep theoretical background, hoping that new finding will urge us to evolve our implementations.

Towards this direction, we take the courage to feel free enough to pose our bet for the winner of inertial navigation race, and our guess is that inertial sensor manufacturing companies themselves are the best candidates until now. This belief rises from the fact that back to the roots of inertial guidance, the manufacturers were the first to achieve it and the ones to define the elementary theoretical framework in the field. On the other hand, due to the project's significant difficulty of accomplishment, the sensor suppliers are more likely to be forced to align with third party map providers and mobile phone manufacturing industry. In plain words, we believe that inertial navigation if ever applied to a mobile phone, probably will not be a single positioning service provider's asset.

[97] http://www.lib.uth.gr/LWS/el/el_hp.asp

References

[1] Wrigley, W., "HISTORY OF INERTIAL NAVIGATION", NAVIGATION, Vol. 24, No. 1, Spring 1977, pp. 1-6.

[2] Sensor Fusion on Android Devices: A Revolution in Motion Processing, David Sachs, Google Tech Talk, August 2, 2010 http://www.youtube.com/watch?v=C7JQ7Rpwn2k

[3] Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration, The International Journal of Robotics Research January 2011 vol. 30 no. 1 56-79

[4] Kourogi M. and Kurata T. (2003). "Personal positioning based on walking locomotion analysis with self-contained sensors and a wearable camera." Proc. 2nd IEEE and ACM Int. Symp. on Mixed and Augmented Reality, pp. 103-112.

[5] Galindo, C. et al. (2005). "Vision SLAM in the measurement subspace." Proc. IEEE Int. Conf. Robotics and Automation, Barcelona, Spain, pp. 30-35.

[6] Inertial Navigation Versus Pedestrian Dead Reckoning: Optimizing the Integration, Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation

[7] Personal Dead-reckoning System for GPS-denied Environments, Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on

[8] Multi-resolution SLAM for Real World Navigation, Proceedings of the 11th International Symposium of Robotics Research

[9] A Zero Velocity Detection Algorithm Using Inertial Sensors for Pedestrian Navigation Systems, Sang Kyeong Park and Young Soo Suh

[10] Improvement of indoor positioning with inertial sensors by context sensitive algorithms, U. Walder*, T. Bernoulli, and T. Wießflecker, Graz University of Technology,

Institute for Building Informatics, Graz, Austria, 8010

[11] Pedestrian localization for indoor environments, Dissertation submitted for the degree of Doctor of Philosophy, Oliver J. Woodman, St Catharine's College September 9, 2010

[12] Indoor Navigation Performance Analysis, Proceedings of the European Navigation Conference GNSS 2004

[13] Indoor Navigation System for Handheld Devices, Worcester Polytechnic Institute Worcester, Massachusetts, USA

[14] Redefining Einstein's Velocity Addition Formula, 2010, Joseph A. Rybczyk