Department of Computer Science and Engineering
The Chinese University of Hong Kong

LYU 1204
A Mobile Assisted Localization Scheme for Augmented Reality

FYP report spring 2013

Tsang Sze Hong (1155000577)

Supervised by

Prof. LYU Rung Tsong Michael

Abstract

Today's mobile devices such as smartphones and tablet computers not only possess high computational power, but also equipped with a variety of measurement sensors such as accelerator and magnetometer.

This is a final year project in a group of two persons. In this project, we utilized different sensors of mobile device to measure indoor distances and aspect ratio so as to generate a 2D floor plan of an indoor environment. With the floor plan generated, user stands on an arbitrary position of the same room and capture a corner can update his current position on the floor plan.

In the first term, the major achievement is the on the part of measurement and floor plan generation. In the second term, the major achievement is the improvement on accuracy of the sensor reading, updating device position and building an app demo showing all the function.

We have studied some relevant topics of the research. Then design and implement our system. We have done experiments and testing on the accuracy and usability of the system and tried to improve.

Chapter 1 of this report will give a brief introduction of this report.

Table of Content

1. Introduction

In this report, **Section 1.3 will briefly give overview of the project.** Chapter 2 will report on the relevant topic we have studied. Chapter 3 will report on our development background. **Chapter 4 will be the detailed design with Section 4.1 as system structure and Section 4.2, 4.5, 4.6 and 4.7 will be the major work done in the second semester.** Chapter 5 is the guidelines for using the application. Chapter 6 reports on all the experiment we have done with **section 6.9 will summaries all the important conclusions**. Chapter 7 will be a complete report on some difficulties encountered. Chapter 8 will be the concluding chapter with **section 8.1 summaries the work done in both semester.** Chapter 9 is the table of division of work. Name of other chapters can be found in table of content.

1.1 Motivation

In this chapter we will write about how the project idea comes out. In these years, mobile devices like smartphones and tablet computers had become necessities in everyday life and cannot live without these devices. And they are equipped with advanced and powerful measurement sensors like magnetometer, accelerometer, gyroscope and compass etc. These sensors can help to measure and capture the status of the device and the environment around it easily. It is interesting to utilize these sensors.

Global Positioning System (GPS) are incorporated into mobile devices. Receiving signal from the satellite in the space and obtain the current location information of the device. It is useful and accurate enough navigation purpose.

However, GPS is restricted to outdoor usage only. Indoor area is difficult to receive the satellite signals and cannot be use for indoor navigation purposes. But with assisted GPS(A-GPS), mobile phone and make use of network data for indoor positioning and navigation. For example, Google maps 6.0 started to support indoor positioning and navigation function. Although indoor positioning and navigation technology is evolving, the covering on these issues is will very little. Nokia indoor navigation system “Destination maps” included only 4605 buildings. We can see that technology for indoor environment still have possibility for improvement. Therefore we decided to do something indoor.

Also, Mobile devices are having very great computational power. Many have multi-core processor, gigabytes of memory which can compete with personal computer. Rendering 3D scene is no longer a problem for these devices. Many complex 3D games with beautiful graphics can be run on these devices.

With the above observation, mobile devices are definitely more than just for making calls and manipulating documents. Thus we wanted to utilize the sensors of mobile device to capture the status of indoor environment and recognize the current position as a localization scheme. This brought us the project idea.

1.2 Background

Current technology gave us insights to further extend our project idea and developed into this project work. We have reviewed some software and mobile application in the market. The followings would be brief description as well as some comments about the software that brought us insights.

IKEA Home planner

IKEA Home planner is a 3D tools developed by the furniture company IKEA, running on PC or Mac computer browsers. This is a tool for costumers to design their home with new furniture put on virtually. It provides 2D floor plan view and 3D drawer functions for users. In the floor view, users can add actually measurement of features in the home such as wall length, door to draw the update the floor plan. The 3D view function gives costumer a more realistic plan for the home. User can draw and build a 3D view of the home and place the product of the company on it. The software finally records the product information and prices for costumer as reference.

This is quite useful software for users who move to new home but we still see some inconvenience in this software. First, it is not portable as it is on computer and the installation is not trivial. Having a version on mobile device would be more convenient. Also, user needs to enter every length to the program to generate the view which is rather troublesome. So if somehow we can get the ratio of different lengths automatically by using the mobile device to capture scenes from the indoor environment and enter one length for calibration, it would be much more convenient.

MagicPlan

MagicPlan is an iOS application for ipad on itunes. Its can measure user's room and build a floor plan by the using camera to obtain the indoor scene. Floor plan can be generated in different format such as PDF or JPG. It can also publish an interactive floor plan on web to let you build home virtual tour to walk around in your home. MagicPlan build floor plan by first starting the camera and let user to mark each corners in the home. When all corners are marked, the floor plan can be generated.

MagicPlan is a user friendly application which can be used without much limitation. It's user interface is easy to use. The measurement and floor plan generation function gives us insights on our project development. However, it is a commercial product but not an open source project, we cannot look at the implementation. We would use it as a comparison agent to our product. Also, it is an iOS application only and doesn't have an Android version.

Floor Plan Creater

When it comes to Android platform, there is also an app named Floor Plan Creator which allow user to build a floor plan interactively. Users can choose to draw the floor plan on the blank page or interactively build it by marking the corner of the room like the MagicPlan.

However, this one is much more difficult to use than MagicPlan for iPad. It is not easy to creator a floor plan successfully and the app is always stuck on some point and cannot move on. Clearly the iOS version for floor plan creating

application is more advanced and nicely built. Thus this would be part of the reason we choose Android as our development platform.

Smart tools

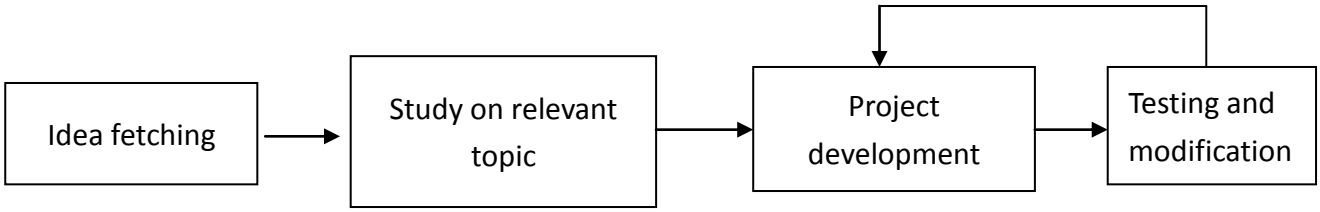
Besides floor plan generating applications, we have also looked into some apps for length measurement. Smart tools in popular in Google Play store with high rank and download times. It is a tool for different measurement as a virtual ruler or protractor. However, the mechanism is just printing markers on the screen like a real ruler. It is not interactively measure the length. It do possess a distance measurement function interactively measure the distance from a object with the input of the height of the object. This is done by some geometric method which would be discussed in later section. We think that interactively measuring length and distance would be a convenient and interesting function.

Advanced Ruler Pro

Advanced Ruler Pro also uses camera as a interactive ruler like the distance measurement function of smart tools. Basically it start with calibrating one of the length of a know object, which is also similar to the procedure of smart tools. The control of this app is quite complicated so there is some room for improvement.

1.3 Project Overview

This section will report a brief overview of this project. After we have insights and motivation described in previous part, we studied some of the relevant topic to see the development of the technology and research. The topics we have studied will be discussed on chapter 2. After review the related topic we started to design and implement our project. While we were implementing our project, we did testing to check the accuracy and tried to improve it if it was unacceptable. The work flow can be summarized by the following flow chart:



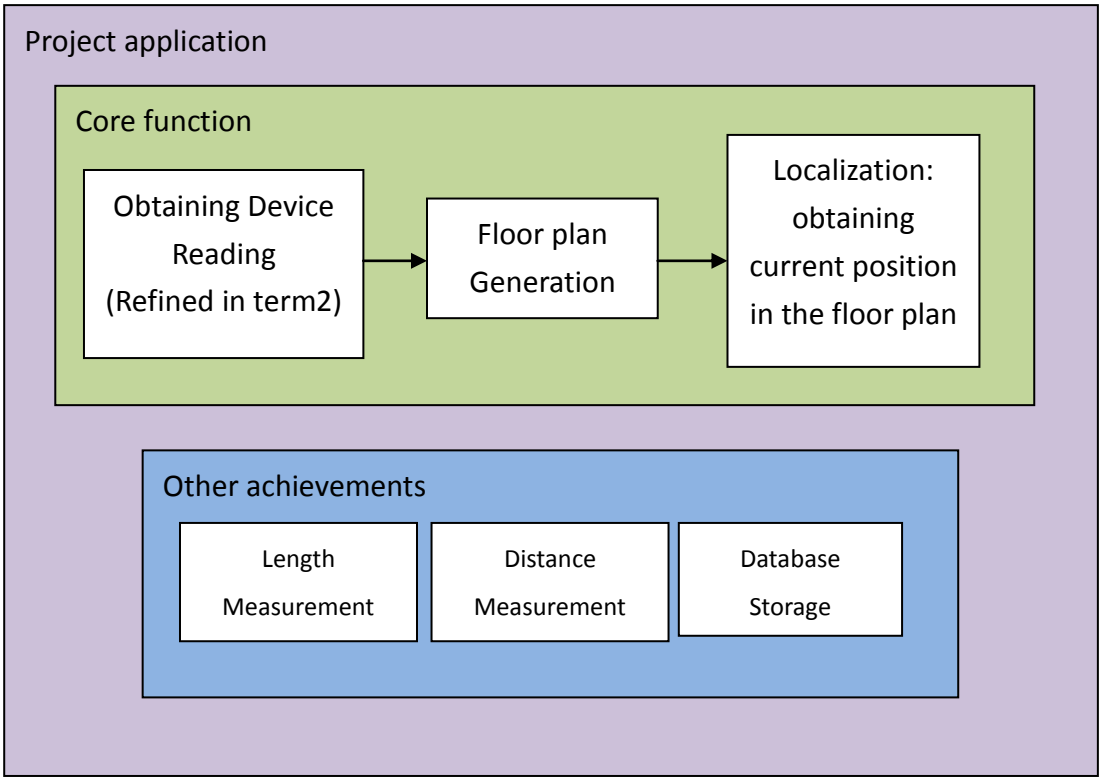
Project development

Our project was to develop and Android application with the **main target to find the position of the person on a generated indoor map, that's the localization part**. First, user stands on an arbitrary position in an indoor room capture different corners. **Sensors reading obtained can be used to compute length between corners and the aspect ratio of a room. Then a floor plan of the room can be produced. If the user moves to another position of the room and capture the first corner, the position on the floor plan is updated.** Additional functions of the application which act as side product was length and distance measurement function. They are used to measure the length and distance of two points or distance from the device to an added point. We also added data storage function to save down the data for later reuse.

This project lasted for one academic year. In the first semester, we have completed the following issues. First, we obtain the device compass reading for obtaining device orientation relative to the environment. We got familiar with the development platform and API to be used in this stage. Second we used the reading obtained to measure length of two points, with some estimation and assumption added for calculation. We did some experiments to reveal the errors of these measurements. Third would be our floor plan generation. We used the reading obtained to generate the aspect ratio of the floor plan and use the length calculated for scaling. In this part, the floor plan generated will have certain restriction because the algorithm used is based on some assumptions. Also we can mark the current position on the floor plan.

In the second semester, there are two major achievements. The first one was changing the method of obtaining the sensors data for manipulation to improve the sensors' reading reliability and enhance the accuracy. We did another set of experiments to see the result. The second achievement was the localization, with the floor plan generation procedure implemented in the first semester, the location on the floor plan can be updated if the user moves to another point on the room and capture another set of readings. There are some minor achievements including separated the measurement functions, data storage and reopen.

The following diagram summarizes the project application



2. Study on relevant topic

In this chapter, we will briefly explain the material we have studied about the relevant topic of our localization application

2.1 Length measurement

We have studied on the different approaches to measure the length of an object or between two points.

From prospective geometry to single view metrology

One way to do length measurement is using prospective geometry. It uses homogeneous coordinates for easier representation of homogenous coordinates. Using the parallel lines can define vanishing lines. Using a know height of object measure the height or length of another object through a know ratio.

Single view metrology is an algorithm summaries previous projective geometry study on measurement. From a view like a photo, it first recognizes some reference lines as axis of perspective and find out the vanishing plane points. Then enter a height or length of a reference object in the photo can obtain the length of the other object in the photo by some mathematics.

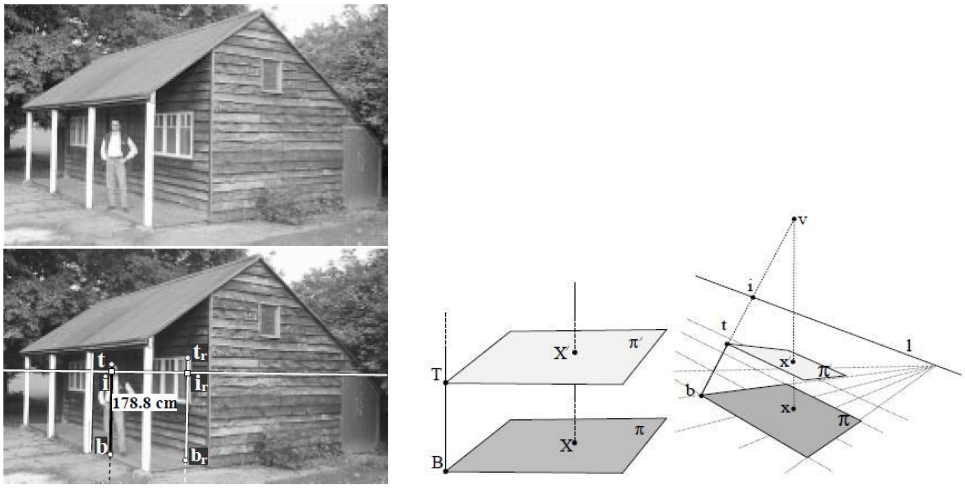


Fig 2.1.1 Single View Metrology and perspective geometry

Although the idea behind is not difficult to understand, the mathematics and implementations of this topic is quite complicated. Thus we choose not to implement our length measurement function using Single view metrology.
[4]

Smarts Tools

In chapter 1, we have introduced the measurement tools "smart tools" on Google Play store. It has a distance measurement function. This app has given some diagram about how is works for us to interpret and understand the background mechanism of this function.

Computing Distance

With reference to fig 2.1.2.1.In order to compute the distance from a target, the system need to have the angle between the ray towards the target and the height of the device from the ground. The height is needed for the user to input. And the target distance can be computed by simple geometry:

$$AB = A'A * \tan(\Theta)$$

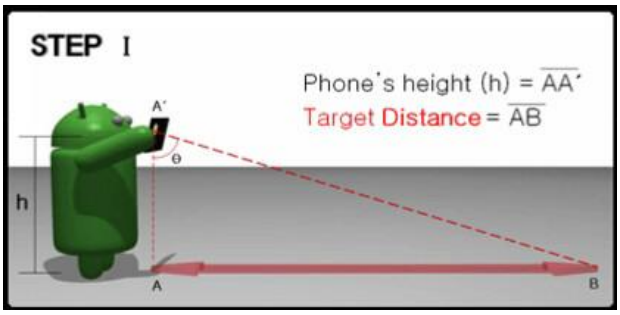


Fig 2.1.2 Diagram in smart tools for distance measurement

If user are standing inside a building and aiming at target out of the building, smart tools allow user to make an adjustment by inputting the height of the building.

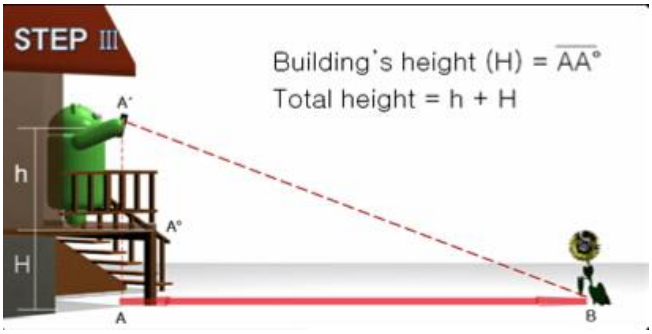


Fig 2.1.3 Diagram in smart tools for distance outside a building

Computing Height

Also by geometric method, the height of a target can be measured by getting the device angle of elevation and height of device from the ground which is inputted by user. With reference to fig 2.1.2.3

BB' = A'A + B'C

And B'C = tan(Θ) * AB

where AB could be obtained the step discussed previously.

Thus, BB' = A'A + AB * tan(Θ)

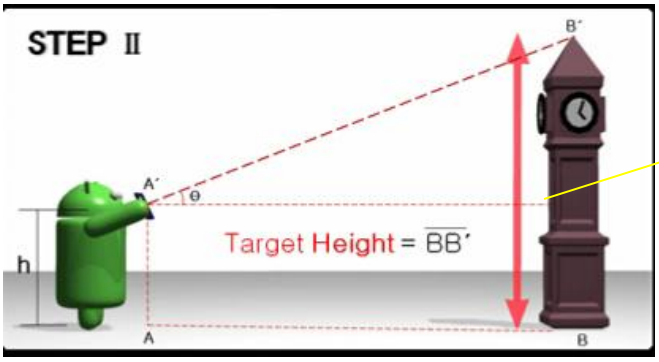


Fig 2.1.4 Diagram in smart tools for height measurement

Computing width

If user rotates along a vertical axis, the angle of rotation can be used to compute the width of an object. With the distance from the user to the left point and the right point, the width of the object can be computed by cosine law:

$BC^2 = AB^2 + AC^2 - 2*AB*AC*\cos(\Theta)$

$BC = \sqrt{(AB^2 + AC^2 - 2*AB*AC*\cos(\Theta))}$

Where AB and AC can be measured by previous steps.

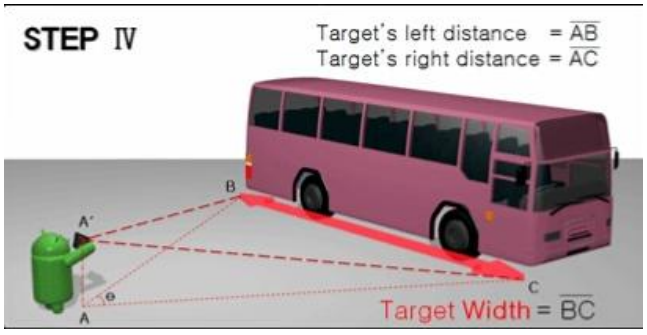


Fig 2.1.5 Diagram in smart tools for width measurement

Thus it can be seen that the distance measurement function of Smart tools works by geometric method. [18]

2.2 Rendering Indoor Scene

Aditya Sankar from the University of has released a research paper on the topic "Capturing indoor scenes with Smartphones". This is about a research on building smartphone application to capture, visualize and reconstruct indoor area. The application first obtains data from smartphone camera, accelerometer, gyroscope and magnetometer to visualize the indoor environment. Then the application can output a real time visual tour for user to explore the transaction of the room.

Interactive tour

User rotates 360 degree to capture a video of the indoor environment. When the capture is completed, the system generates a 360 degree panorama. The interactive play back of room tour is done by indexing the video by camera pose which is obtained from the reading of gyroscope sensor in the smartphone. The instant orientation of the smartphone obtained from the equation:

$$\Theta_t = \Theta_{t-1} + t * w$$

Where t is the change in time and Θ_{t-1} is the previous orientation and t is the change in time and w is the change in orientation. All in all real time interactive tour is done by spatial video indexing.

Floor Plan generation

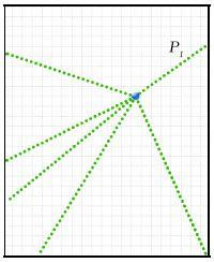
In playing back the interactive room tour, the system allow user to mark the corner. When the marking is completed, the system uses the angle of orientation of the marker to generate the floor plan. For 2D floor plan

generation, it has given the following algorithm:

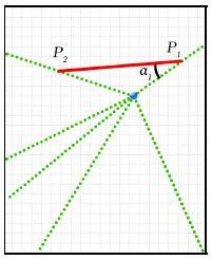
Algorithm 1 Calculating optimal wall configuration

```
P_0 = Origin
P_1 = Unit distance from P_0 along the first ray
N = Number of corners
minDistance = MAX_FLOAT
for α = 0 → 360 step 0.5 do
    for i = 2 → N do
        θ_i = direction of i^th corner                ▷ from (2)
        P_i ← intersect(P_{i−1}, α + (i − 1) × 90, P_0, θ_i)
    end for
    distance = getDistance(P_1, P_N)
    if distance < minDistance then
        minDistance ← distance
        minAngle ← angle
    end if
end for
Return minAngle
```

Using the marker orientation can obtain a series of angle, we can regard it as shooting different rays from the capturing position.



Choose an arbitrary angle alpha and start with first ray, connect a line with angle alpha to the first ray.



Then the following rays are intersected by making a 90 degree with the previous ray.

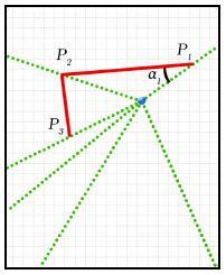


Fig 2.2.1 Steps of Sankar Algorithm

When the last ray meets with the first, the intersection point should be at the starting point if the angle is chosen alpha is correct. However, in practical the two points do not intersect exactly. So the algorithm tries different values of

alpha and chooses the alpha which result in having minimum distance between the last point and the starting point. If the distance d in figure 2.2.5 approaches zero, the first point P1 and the last point P7 merges.

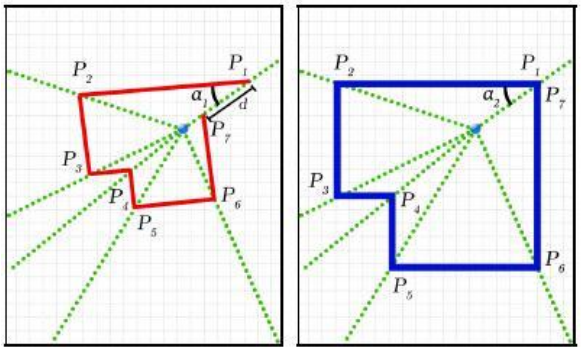


Fig 2.2.2 Steps of Sankar Algorithm

Furthermore, this algorithm is based on assumption that every corner is of degree 90. It is not applicable to the rooms with corners not of degree 90.

Aligning different rooms

After different rooms' floor plan is generated, the rooms are aligned to make the whole floor plan of the home. Floor plans of two different rooms can be connected with the supply of a door ray. Since the door ray of two different rooms are the same, so the two floor plans can be aligned by first multiplying the scaling matrix and then add a translation matrix to the coordinates.

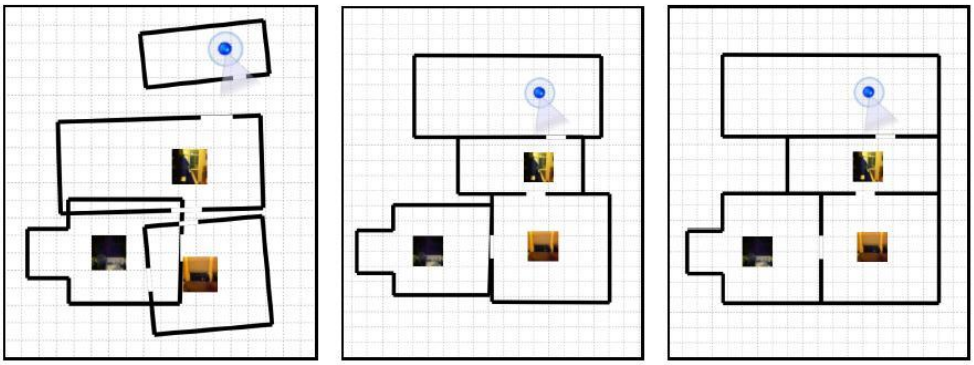


Fig 2.2.3 Aligning different Floor Plans

Considering the complexity of the project and real time interactive visual video tour as it could require a lot of resources, we do not focus on the visual tour. Modeling indoor environment is only one of the tasks of our project. And the floor plan generating algorithm is useful to our floor plan generation function design. We have taken it as a reference to construct our algorithm. We first consider floor plan generation for a single room as our minimum target. If we have time to come back, we will try to include the function of aligning the floor plans of more than one room.

[3]

2.3 Augmented Reality

Augmented Reality is a view of the physical world with elements which is generated by computer. Most AR application can be are marker based. Usually are using the camera to analyze marker recognized or captured. Marker detection involves first converting the image obtained into binary image and then is to compute some points for recognition, called featured points. The last step is identifying the featured points.

Non-marker based AR application are those uses some features such as GPS to display some information augmented by the computer. There are many mobile AR applications and game available in the market.

Our localization scheme product can be used to help in these AR game or applications. One idea is that we can use the floor plan generated as map of the game, which can be used to locate the player and show other information such as monster location in the map. That would become an application aligning the game world and the real indoor environment.

2.4 Indoor positioning

Indoor positioning technology is useful in location based services targeted for indoor environment. It is a connection of wireless network of devices. There are many approaches to achieve indoor positioning. Many of them utilize the in-building wifi network. Using wifi network for indoor positioning is convenient because there is no need of additional hardware component to be installed.

Website Monoj-Kumar-Raja posted an academic project about indoor positioning explaining different method of wifi indoor positioning.

There are two basic methods to find the position of a device with signal strength of the wifi available: trilateration and location fingerprinting. Trilateration is a method can compute the location using signal strength and position of wifi signal transmitter. It is based on an idea that the signal strength is inversely proportional to the signal strength obtained. But this method does not work well in indoor environment since interference of the wall-like equipment may cause a sudden increase or decrease of signal strength gives unwanted result. The other method involves two phases. Signal strength from wifi signal transmitter is first recorded into the database. With more data collected, a pattern of interference can be formed and be differentiated. In the second phase, it tries to match the obtained signal with the training data collected in phase one.

There were experiments on the location fingerprinting method described above. It was done by Bahl and Padmanabhan and Pang. The comparison of their results found that a method named Naïve Bayes method based on calculating

probability density function of the signals from user location and the location is obtained by the highest probability. It was better the nearest neighborhood algorithm in which the location is computed from matching the point of nearest neighbor in the signal. The distance from neighbor is computed by least square between new testing signal and training signal recorded.

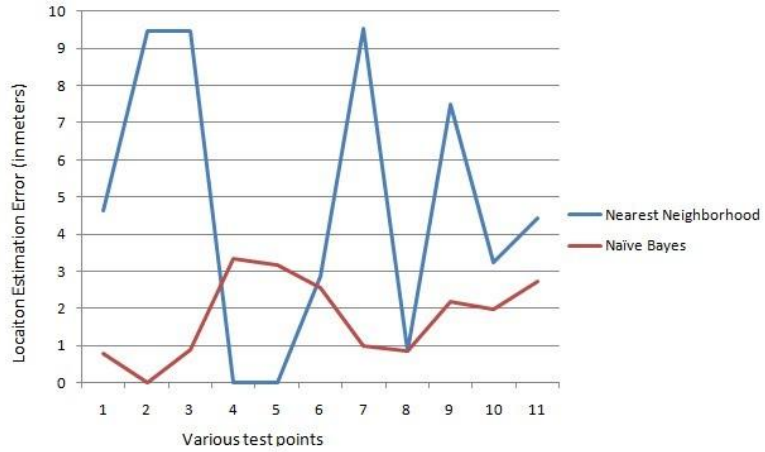


Fig 2.4.1 result on comparing two indoor positioning algorithm

As lots of researched have been done already in this area, we would like to find other ways to do the positioning.

[17]

2.5 Improving sensor reliability using sensor fusion

Background

In the first term, we used the SensorManager.getOrientaion() method from android API to obtain the orientation angle of the device. The angles are based on accelerometer and magnetometer reading. Accelerometer is responsive to gravity influence and magnetometer work like a compass to get the angle. However, from experiment done, the magnetometer reading was very inaccurate with a lot of noise. Researcher Paul Lawitzki posted a method of sensor fusion to handle this issue.

Gyroscope

Gyroscopes inside Android devices have a more accurate reading and shorter respond time. Gyroscope is a sensor which provides angular rotational speed along the axes of orientation. In order to obtain the angle of orientation, the angular speed read from the gyroscope has to be multiplied by time interval between the last and the current sensor output. This would compute the rotation increment at the time interval. Summing up the all the rotation increment gives the orientation. However there are gyro drift causing problem of gyroscope. When there is a drift in the device, the gyroscope will have an offset bias which tends to correct drift. The gyroscope will never become perfectly zero since the gyroscope will not become stationary.

Sensor fusion

The problem of magnetic field sensor is the noise and the problem of gyroscope is gyroscopic drift. To avoid this two problems, gyroscope reading is used only for

short period of changing orientation. In order words, the problems have to be solved by low-pass filtering of magnetometer and high-pass filtering of gyroscope reading. Low-pass filtering of magnetometer means to average orientation angle over time and within a constant time window. High-pass filtering means to replace the high frequency component from magnetometer with the corresponding gyroscope reading. This would be the idea of sensor fusion. A similar approach can be applied to accelerometer to reduce it noise error.

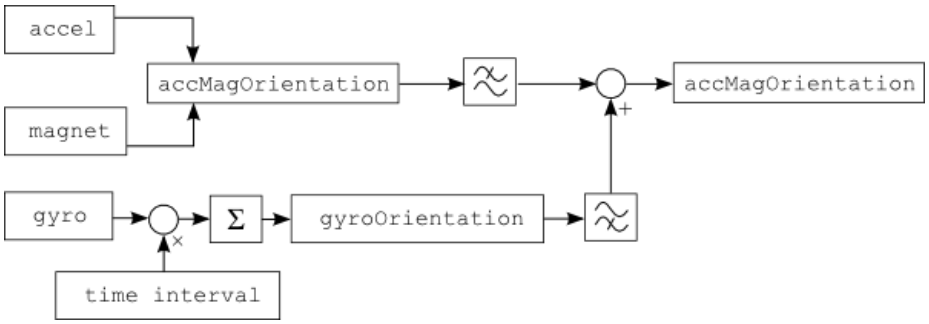


Fig 2.5.1 Work flow of sensor fusion

The simple case demonstrate the idea is as follows: when the device is turned 90°. From the magnetometer fluctuation and gyroscope drift have been fixed after the low-pass filtering of magnetometer reading and high-pass filtering of gyroscope.

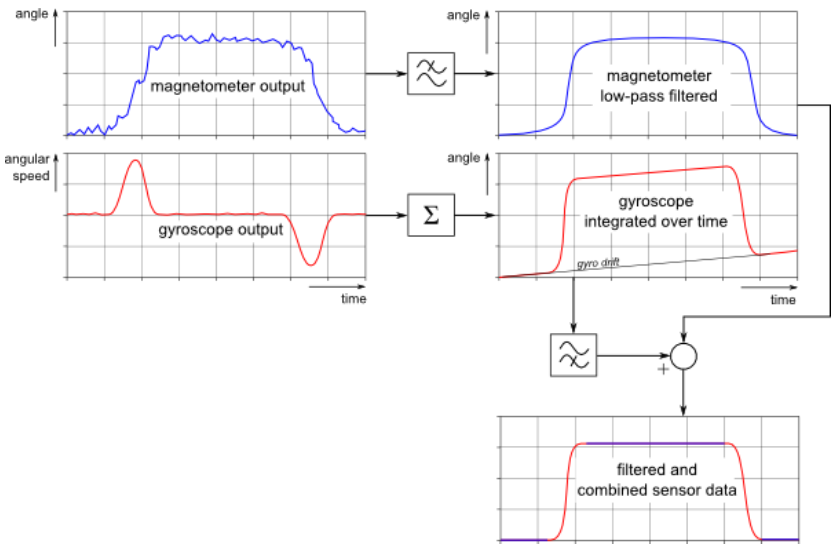


Fig 2.5.2 Result of sensor fusion

This topic of study was useful in helping us to reduce the problem of sensors error. We had chosen to try to adopt the new methodology of sensor fusion to obtain the device orientation for different parts of computation of system. Implementation and experiment was performed afterwards to test on the accuracy of the new method. They will be covered later in this report.

[14]

3. Project Setup

This chapter will report on the environment settings of the project development.

3.1 Target platform

The target platform of our project is on Android platform for tablet or smartphones with sensors accelerometer, magnetometer and gyroscope. The application also requires a camera.

Platform requirement:

OS	ANDROID version 2.3 or above
Hardware	(1) Sensors : Magnetometer , Accelerometer , Gyroscope (2) Camera
Device	Tablet or Mobile Phone

3.2 Development environment

We uses eclipse IDE with Android SDK plug-in as our development environment. Detailed version id number of the software used can be found in appendix. Implementation is in java in which we are quite familiar with. Of course we still need to read the API as some Android API are different with ordinary Android API.

3.3 Device used

In first semester, our development device is ASUS Eee Pad Transformer Prime TF201. The system ran smoothly without crashes on it. Transformer prime is a quad-core tablet with high computational power. Other weaker devices may

run the application slower than Transformer prime. It gets slower when we have used opengl plug-in in our project. Generally, transformer prime computational power fits our program need.

In term 2, we changed to Xiaomi 2 since Transformer prime cannot be allocated to us anymore. Xiaomi 2 is a qua-core smartphones with the sensors we needed. We were comfortable developing the application with it. And the program ran without lagging.

The application should also run in other Android devices with above requirement without problem but since we do not have other.

4. Detailed design

In this section, we will explain our work in a detailed manner. **Section 1.3 gives a better overview of the work in brief idea.** We will describe the structure of the system and then describe each function in detail.

4.1 System Structure

The structure of our system can be classified to three main categories, View, Controller and Model. View handled what would be displayed to the user interface from the model. Control communicates with the device input. MODEL process the main functions and structure of our application such as the computations and main algorithm. In each of the categories, there were several modules with specific function. All together there are 17 modules.

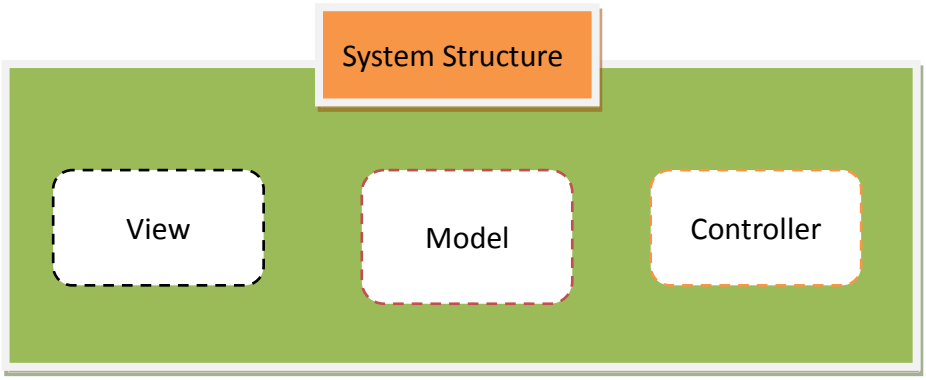


Fig 4.1.1 Structure of our system.

View

View handled the items need to be displayed in the user interface. In View, there are 2 modules:

CameraPreview: CameraPreview module used API provided by Android to call the camera and display its view out to the screen and contributed to a layer in the user interface.

DrawView: DrawView module was responsible for another layer of the user interface. All the words and control buttons are handled in this module.

OpenglRenderer: OpenglRenderer rendered the floor plan out to the screen. This module set the attributes such as background and color of the gl world. The code written was with reference to the some opengl es standard tutorial listed in the reference list.

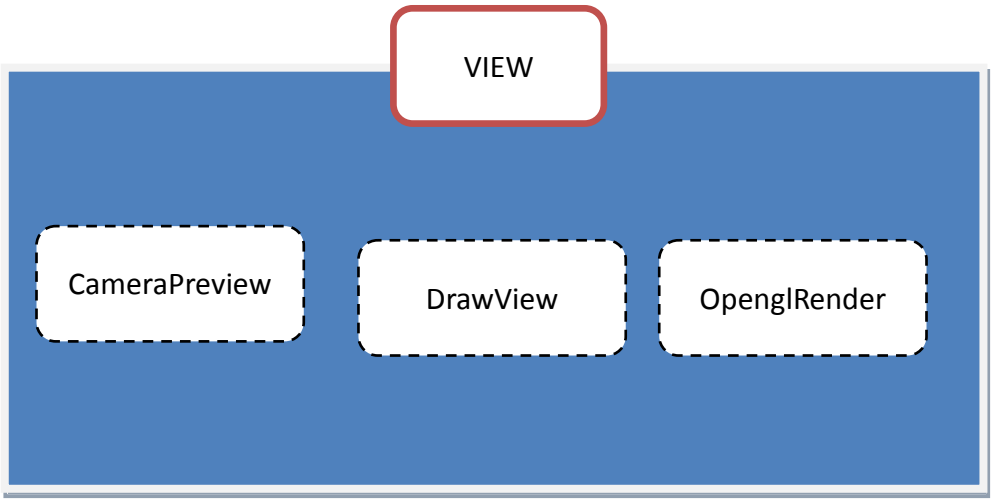


Fig 4.1.2 Structure of View.

Controller

Controller is the categories for handling the input of user, which included the press on button event, and all kinds of listener. This is a small category consist of only one module, which is the main android activity of the system:

CameraTestActivity: CameraTestActivity is the main android activity. It handles the users input such as pressing the different buttons.

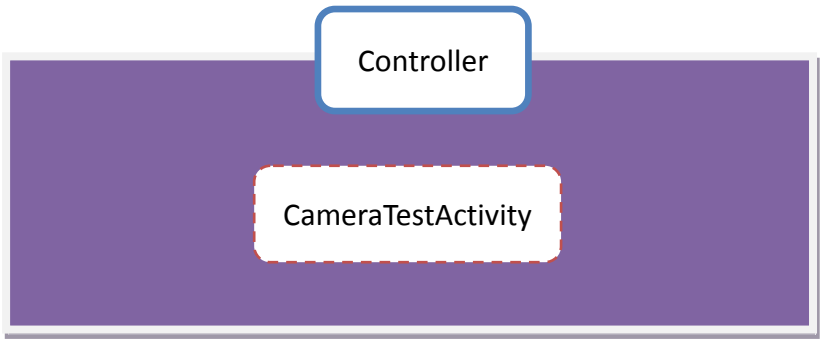


Fig 4.1.3 Structure of our controller.

Model

Model handled all the functions and algorithm to be performed. It is the core part of the system such that all the important data communication and manipulating is in this categories. The function of each module is showed in the following:

Accelerometer: Call the accelerometer to get the reading from it.

Compass: Call the compass to get the reading from it.

Gyroscope: Call the Gyroscope to get the reading from it.

DataStorage: Data communication between modules within the program on run time required the DataStorage module's data. The data reading recorded and floor plan generated is stored in this module.

FloorPlanGen: FloorPlanGen module implemented our floor plan generation algorithm, which will be discussed in detail in section 4.3

Formula: Formula module implement the method of using sensors obtained to do the measure lengths and distances. The method will be discussed in detail

SensorFusion: Sensor fusion is the module for combining sensors data for better measurement. This is the only module which **we used others work**. The module is provided by Paul Lawitzki. Details of the sensor fusion method idea had been discussed in section 2.5 and the uses will be discussed in section 4.2.

UpdatePosition: UpdatePosition module implemented the localization algorithm, which will be discussed in detail in section 4.5

ViewUpdateThread: ViewUpdateThead created a thread to update the text to be displayed on screen such the distance measurement result. Also it created a thread to check whether the first corner is returned during the corner capturing process.

MapDBOject: MapBDOject module stored the all the coordinate to be stored in the database.

DatabaseHandler: DatabaseHandler module handled the data saving for later uses function. It handled all the SQL command for the creating table, store and retrieval of data in the database.

Converter: Converter module provided all the functions and method to do conversion of data. It is needed when the data cannot be directly used.

MyActivity: MyActivity stored the main activity when the running android activities

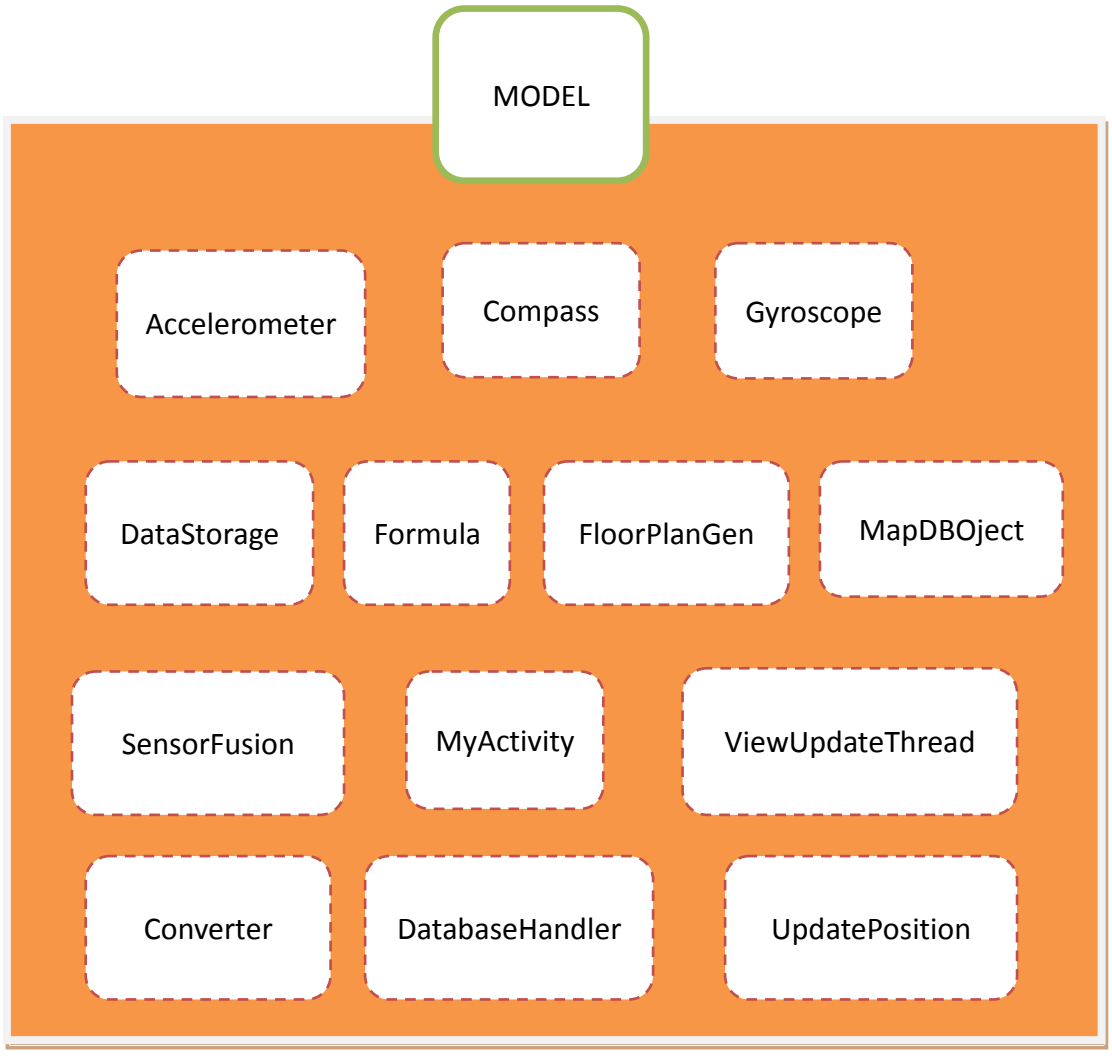


Fig 4.1.4 Structure of our model.

4.2 Obtaining orientation

We need to obtain the device orientation before carrying out the measurement. In the first semester we utilized the accelerometer and compass reading to obtain the orientation of the device only.

Accelerometer class gets the acceleration of the device. The parameter is in 3 dimensions: x, y, z. The compass class reading obtained is in azimuth, pitch and roll. The gyroscope class gets the value based on the principle of angular momentum.

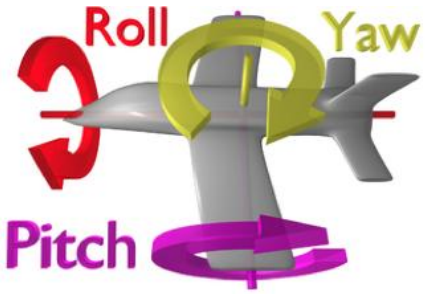


Fig 4.2.1 Pitch, yaw and roll

In semester one we use mainly the compass reading. See figure 4.2.2 the value α can be obtained from the compass directly, it is called PITCH. Also, the value β can be obtained from the compass indirectly, it is called AZIMUTH. The readings are obtained using the getOrientation(float[] R , float[] result) function. Result[0] is azimuth and Result[1] is pitch and Result[2] is roll. They are useful in later on computation.

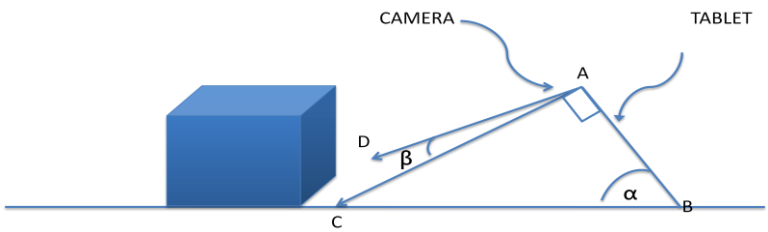


Fig 4.2.2 tablet orientation

However, from the experiments conducted in the first semester show that the error of the compass reading was large. It was because the reading of compass obtained is fluctuating all the time. The reading is not steady and inaccurate due to the noisy nature of magnetometer. We tried to improve the error and found the method in section 2.5: sensor fusion. **We changed the reading obtaining methodology to sensor fusion in second semester.**

Recall that sensor fusion is merging the reading of gyroscope. Paul Lawitzki provided a module for handling the low-pass filtering and high-pass filtering method. We first studied the code provided.

It first got the SensorManager and initialized the sensor listeners. The module implemented the SensorEventListener so some function had to be overridden. For gyroscopic reading, Sensor.TYPE_GYROSCOPE in android documentation provided how to process the gyroscope raw data. And the gyroscope reading is processed when the orientation from accelerometer and magnetometer is obtained. The module converted the sensor reading vector to matrix for storing and processing. Next thing was to eliminate the problem caused by sensor. The module implemented a complementary filter by executing a separate time thread. FILTERING_COFFICIENT is set for calibration. Finally we could obtain the combined device orientation along the axes.

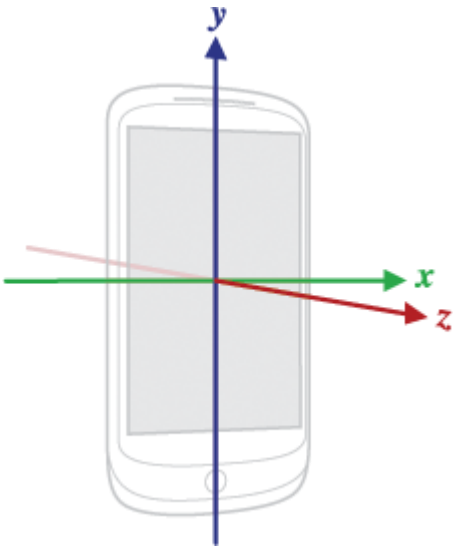


Fig 4.2.3 Device orientation

Thus, We have studied and understood the code provided by the Paul Lawitzki although we had not implement our own sensor fusion module. We have also tested each combination in the sensor fusion and without it. For direct result from accelerometer and magnetometer, use SensorManager in Android API only could help. And finally we used the sensor fusion to obtain the horizontal angle α and β.

4.3 Floor Plan Generation

Next part is a floor plan generation function. User stands on a point inside a room and use the red corner marker on the user interface to target the corner of the room. Then the user presses the capture button to capture a corner. Then the user rotates clockwisely to target at the next corner to capture it. The steps are repeated until all the corners are captured. Then the floor plan can be shown on the screen by pressing the gen button. The output floor plan is drawn in openGL plug-in.

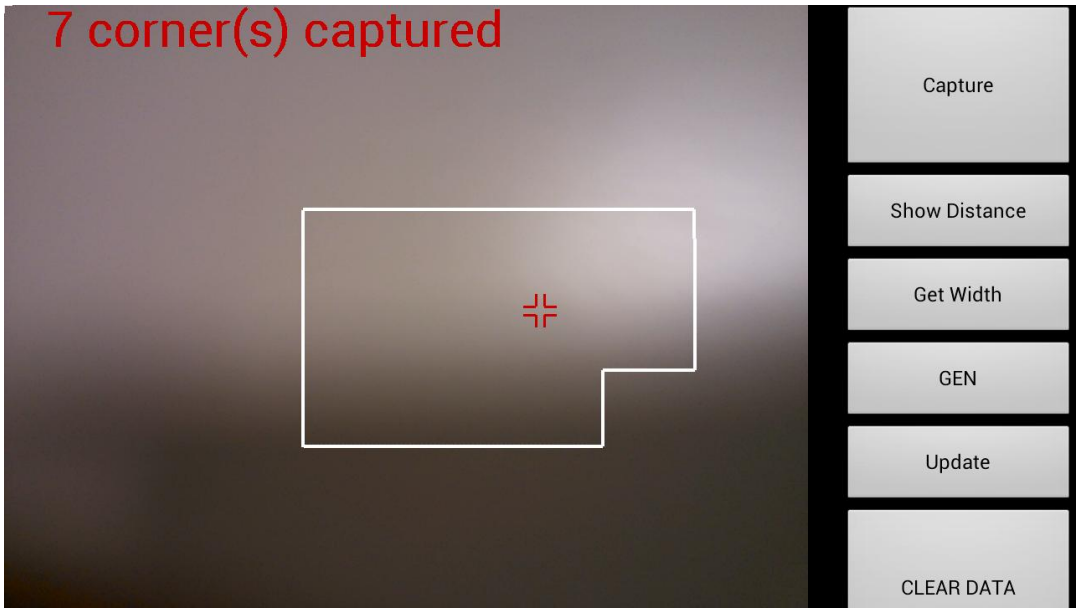


Fig 4.3.1 Floor Plan Display using openGL

We have designed our own algorithm for floor plan generation taken reference to the algorithm of Sankar on Chapter 2.2. The pseudo code of our algorithm is as follows:

Floor Plan Generation Algorithm Pseudo Code

```
minDist←minimum distance from the first to last point
angles[] ← input angle array
output[] ← output array storing the points
for(θ  = 0 → 360 step 0.1){
        sumAngle←θ
        P1X ←tan(θ)/√(1+tan²(θ))
        P1Y ← P1X* tan(θ)
        for(i=2 → no of corner+1 step 1){
                sumAngle← sum of previous angle in angles[]
                if(i%2 == 0){
                        PiX←P(i-1)X
                        PiY←PiX / tan(sumAngle)
                }else{
                        PiY←P(i-1)Y
                        PiX←PiY * tan(sumAngle)
                }
        }
        dist←calDistance (Pi+1, P1)
        if(dist<minDist){
                output←PiX, PiY
        }
}
```

The algorithm inputs an array of angle recorded and outputs an array of coordinates of the generated floor plan. The array is then pass to openGL rendering module.

Our algorithm starts from calculating the first point on the coordinate plane. The line connects from the original to the first point makes an arbitrary angle θwith the horizontal axis. The value of θ will be adjusted by iteration.

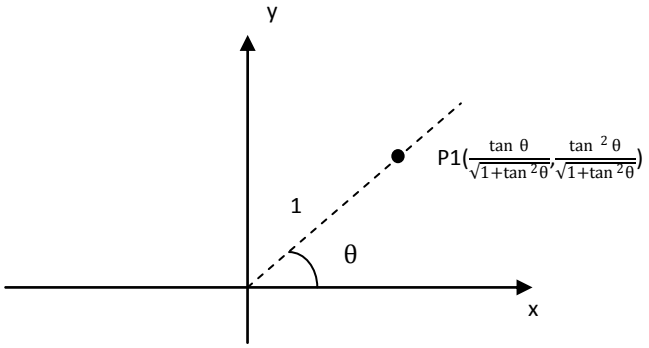


Fig 4.3.2 the first point is computed from an arbitrary angle θ

Next add a ray which makes the angle from input array with the previous line. The next point is the intersection between the horizontal line passing through the previous point and the added ray. As seen in the following figure:

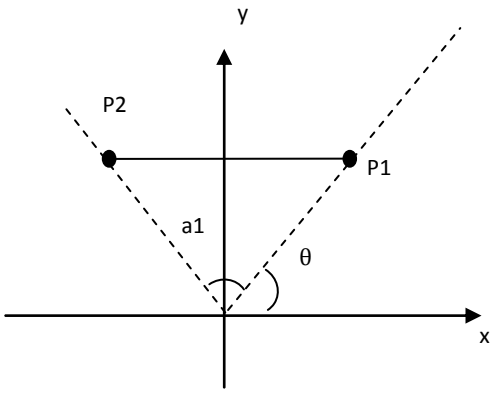


Fig 4.3.3 find the second point by intersecting

Repeat the steps by replacing the horizontal line and vertical line alternatively, i.e. if the ith iteration intersects a horizontal line with the new ray, then the i+1th iteration intersects a vertical line with the new ray. This assumes the angles of all corners in the room are of 90 degree.

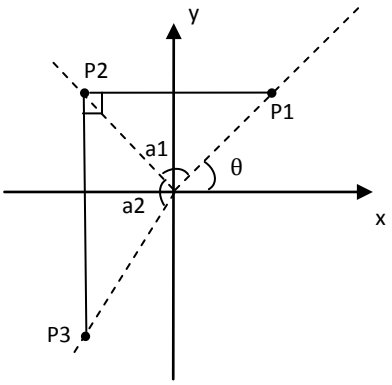


Fig 4.3.4 Assume angle is 90^o to find the next point

Then the steps are repeated until the new added point is on the first ray.

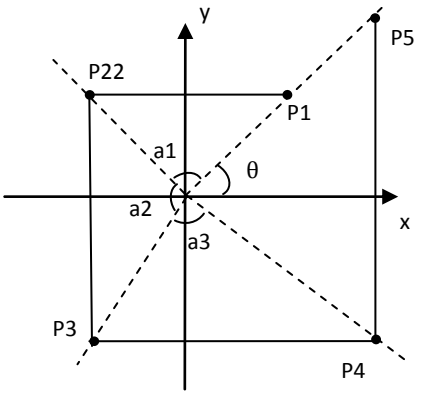


Fig 4.3.5 if unsuitable θ is chosen, the first and last point don't meet

Trying different values of θ in order to minimize the distance between the first and the last point so that they merge together. The floor plan with the minimum distance will outputted.

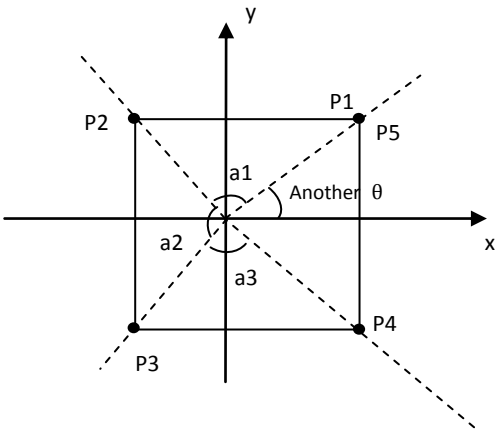


Fig 4.3.6 With suitable values of θ, first point P1 and last point P5 merges

The final θ at the end of the loop will be output to storage for the later use and the use of this value will be explained later.

Of course this algorithm is also based on the assumption that angles of all corners in the room are of 90 degree. It is not applicable for rooms with corner of different angles. Another assumption is that users are capturing the corners inside the room. Testing on this algorithm is in chapter 6.

4.4 Measurement

Next we measure the length between two corners in a room. User can use the red corner marker on the user interface to point at a corner of the room.

Fig. 4.4.1 Aim at a corner

When user touches on the capture button, the system will record the compass reading of the device. And the reading will be shown on the screen. Next, user uses the red corner marker to point on the next corner, and touches on the capture button again. Then the measured length will be shown on the screen at the upper left corner.

Fig. 4.4.2 Length shown on screen

We use geometric method to compute the length. The step is as follows.

Step 1:

When user capture the first corner, the orientation of the device is obtained. We record the angle$\theta$the device makes to the horizontal.

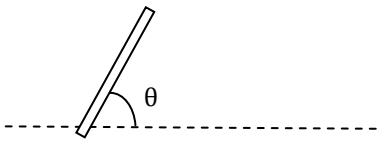


Fig 4.4.3 Recording the angle that device makes with horizontal

Then we estimate the height of the device above the ground is as 1.4m. With this assumption we can formulate the vertical distance from the device to the target corner as follows:

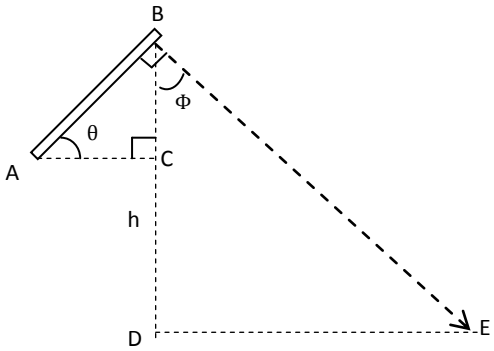


Fig 4.4.4 formulating distance from device to target corner

AB represents the device holding by the user and E represents the corner of the room the user pointed at.

$$\Phi = \angle ABE - \angle ABC$$

$$= 90^{o} - \angle ABC$$

$$= 90^{o} - (180^{o} - 90^{o} - \theta)$$ Since (Triangle ABC is right-angled)

Since θ have obtained already and h as estimated as 1.4, we can compute the

distance from the corner as:

$$DE = h * \tan(\Phi)$$

$$= h * \tan(\theta)$$

Step 2:

When the user points at another angle can press capture, the system record another set of compass reading for to get the device orientation. Moreover, the angle β is recorded.

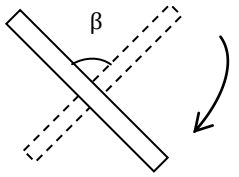


Fig 4.4.5 top view of rotating the device to point at another corner

With β the length the first captured corner to the second corner can be computed as follows:

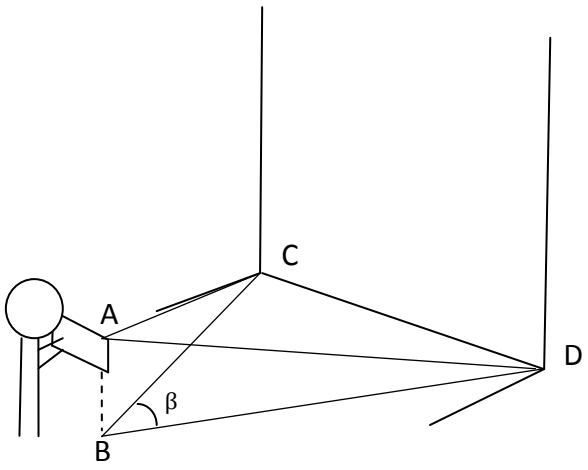
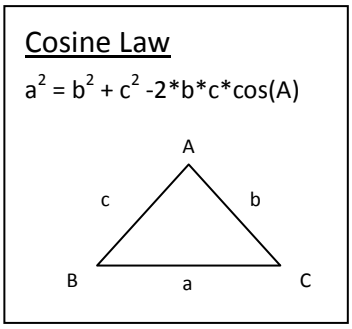


Fig 4.4.6 measuring distance between two corners

The user wants to measure the length between the corner C and corner D of his room. He first aims at C to capture the first corner and rotate to aim at D. The angle β is assumed to be the same as angle A. Then length between CD can be

found:

$$CD = \sqrt{(BC^2 + BD^2 - 2*BC*BD*\cos(\beta))}$$



BC and BD is obtained with the method explained in step 1.

There is a minor assumption that the angle obtained is roughly equal to β since it should be the angle $\angle CAD$ instead of $\angle CBD$.

4.5 Localization

With the floor plan generated, the next main function is updating the current position. To explain that we have to start by discussing how did we record our floor plan generated. From the floor plan generation algorithm, we obtained an array of coordinates representing each corners of the room and the corners are arranged by the sequence when it is recorded. At the original (0,0), it was where the user stands on and capture the device orientation. Also recall that we have saved an angle θ which is the angle between the horizontal and the line connecting the origin and the first corner. Also the distance from the original to the first corner is 1. The information available can be represented by the following diagram.

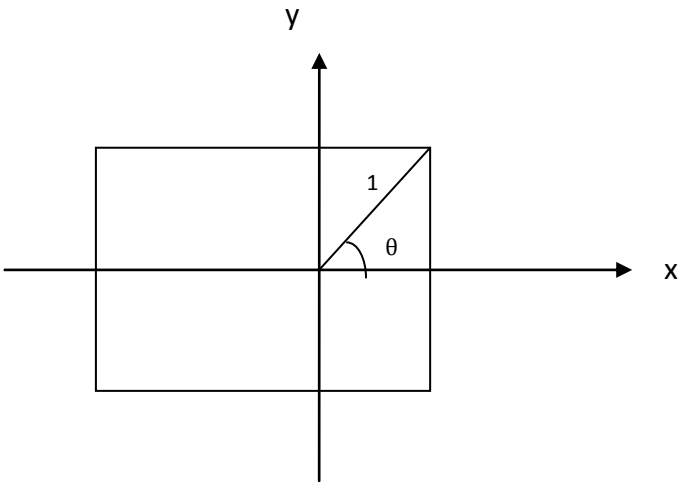


Fig 4.5.1 Coordinate system when floor plan is generated

Consider when the device is moved to another position of the room:

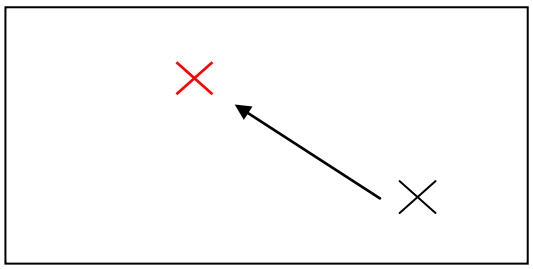


Fig 4.5.2 Moving of device in top view

Since the direction is important, for convenient calculation, we set the x direction as the N 0^o for the coordinate system of our floor plan so that the angle value is fitted to our coordinate system.

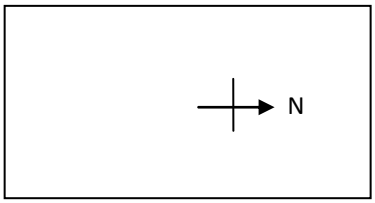


Fig. 4.5.3 Proposed coordinates system with angle

However, in most of the circumstances, the actual magnetic orientation would not align with our coordinate system.

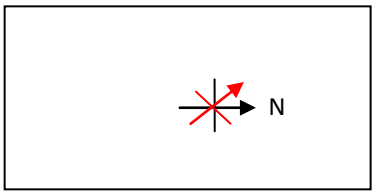


Fig. 4.5.4 Overlapping with our coordinate system

Transforming the room orientation by an angle offset, the room can fit our coordinate system.

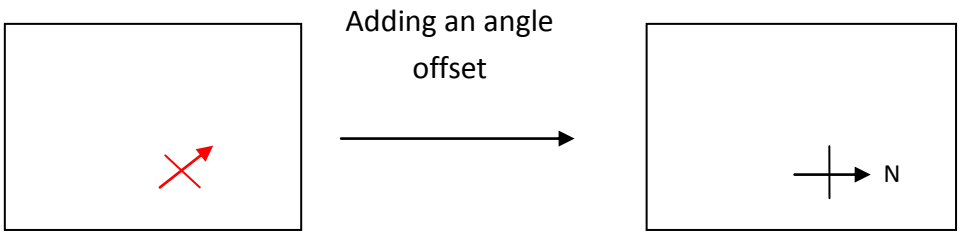
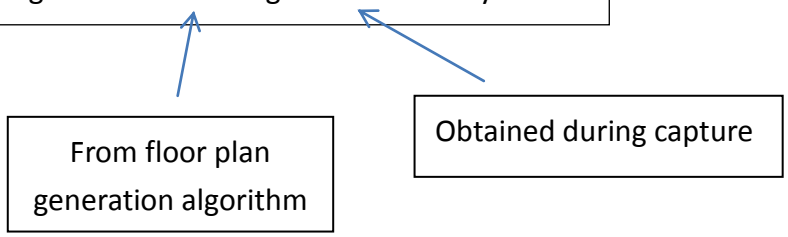


Fig. 4.5.5 Transformation of the system

The angle offset here is the different betweenθ stored when the floor plan is generated and the compass angle obtained when capturing the first ray. We also need the distance from original to the first corner during capturing process as a scale to the length “1” in the coordinate system.

That is, we need the following before the localization can be done:

1. Angle Offset =θ – angle of the first ray



2. Real length to coordinate Scale -- record length from the first corner captured.

With the information above pre-recorded during capture. The update function can be performed. We have designed two ways to do the update procedure, each with its own restriction.

The first one require user capture only the first corner obtained this method is used when the number of corners is larger than 4. The system will pop up automatically to tell user only capture the first corner and will assist the user. When the user capture the required corner, compass reading and the distance from that corner will be recorded.

The compass reading obtained will first transform into our coordinates system and scale down the distance.

Since we have the coordinates of the first corner (x1,y1) record as the output of floor plan generation, the picture of our coordinate system will become the following:

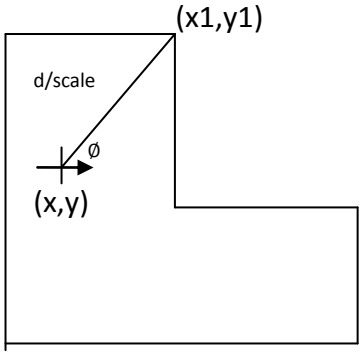


Fig. 4.5.6 Coordinate system

Since the distance from the corner is fixed and we have the angle pointing to it, we can compute the target x, y.

$$x = x1 - h*\cos(\emptyset)$$
$$y = y1 - h*\sin(\emptyset)$$

We need to restrict user to capture only the first corner because we need to locate the corner at which the user is pointing at, which cannot be found easily.

Another way does not require user to capture only the first corner allow arbitrary corner target. That is the user can aim at any corner inside the room to update the position. However, this is restricted to 4-corners only. We did this by classifying the compass angle (transformed to our coordinate system) capture into four quadrants, each quadrant corresponds to one corner. Then the corners which were aimed by the user can be located automatically. When it is located, the coordinate of the new position can be computed using similar method above.

Frankly, we have not found a way to automatically support all corners and number of angles greater than 4.

4.6 Database Storage

The system supported save-and-load function for the floor plan generated. At run time, we store the data in the data storage. The data required to be stored down was the coordinates of corners, angle offset for transforming coordinate system, the distance scale from the origin to the first corner. They would be used to render the floor plan out again.

We implemented using SQLite for Android, which is a file based database build in the android. The database of our program is rather simple since only the few attributes were stored. One map storage table is needed.

4.7 User Interface

In the main UI design, there were 3 layers, namely openGL, camera preview and drawing view layer. We have also a menu which specially deal with the save and load function.

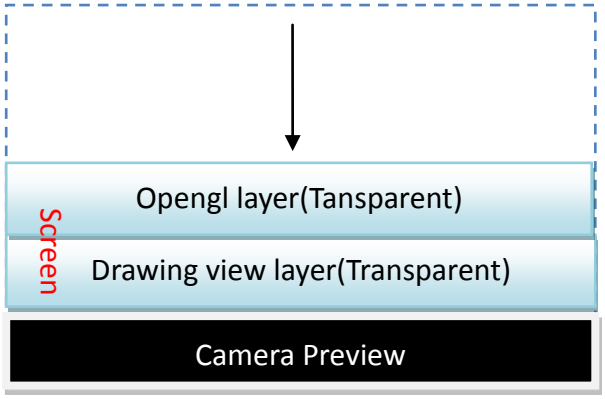


Fig 4.7.1 User interface layered structure

The bottommost layer is the camera preview layer. It was used to display the view of the camera to assist the user to aim at a corner.

Fig 4.7.2 User interface layered structure

The middle layer is the drawing view layer where all buttons and words are displayed out in this layer. There is a red cross in the center of the camera view in this layer which is used to assist user to mark on the corner. The button control of our application is also drawn in this layer. The button is used as input of the different function. The detailed uses of the buttons can be found in section chapter 5

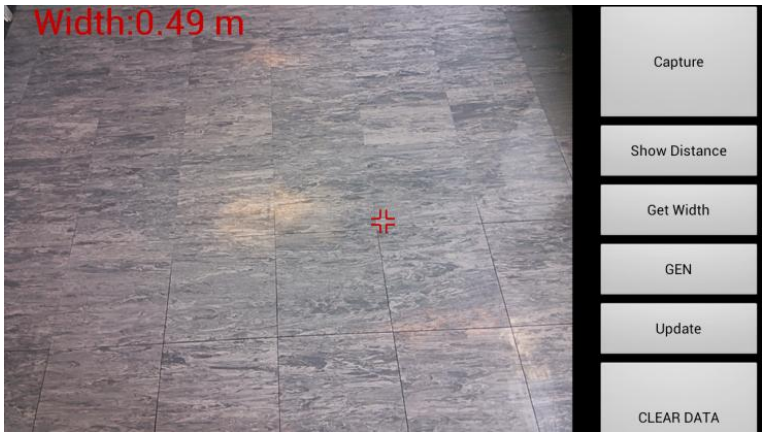


Fig 4.7.3 Drawing view layer

The topmost is OpenGL layer. We use android openGL es to render out the floor plan on from our coordinate system. The openGL layer is full screen and the background is set to transparent so as to ease the user when updating the position. Also we needed to scale up the map for fitting.

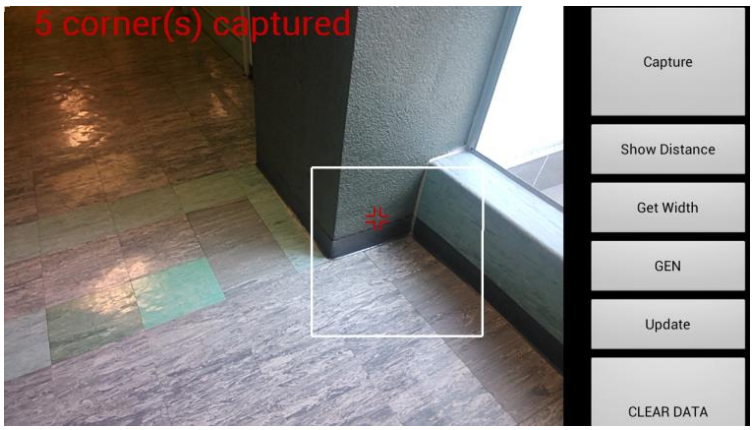


Fig. 4.7.4 opengl layer

There were differences between openGL and our original. The coordinate of openGL system is starting at (0,0) at the upper left corner, increasing x and y along the right and down direction. But our coordinate system having original at the position which the room is captured, but increasing at right and up direction. Since our coordinate system cannot be directly mapped to the openGL coordinate system, we needed to do some conversion.

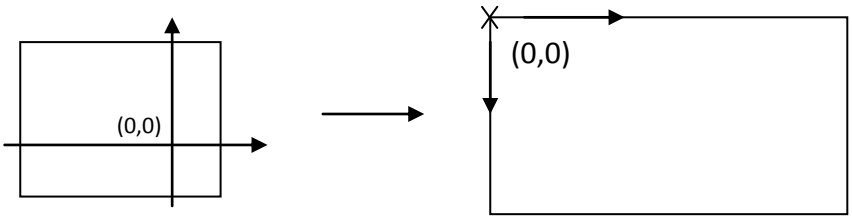


Fig 4.7.5our coordinate system change to opengl coordinate system

The save and load back to database is being performed in a menu bar. The menu pop out when upon the android menu button is clicked.

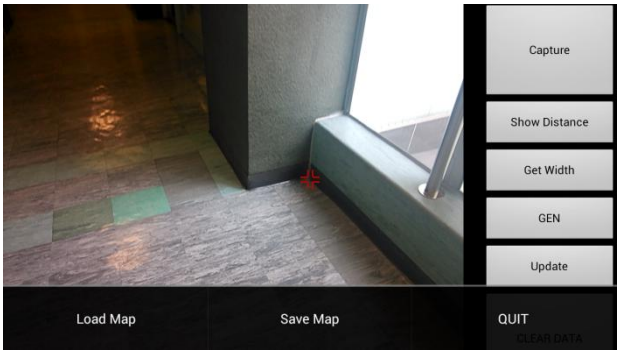


Fig 4.7.6 menu bar

We have changed our design in semester one. The main UI was having 2 layers instead of 3 and the floor plan was rendered in a separated activity using openGL. We add one layer for easier handling of the localization function. Menu bar was added in semester two for data storage.

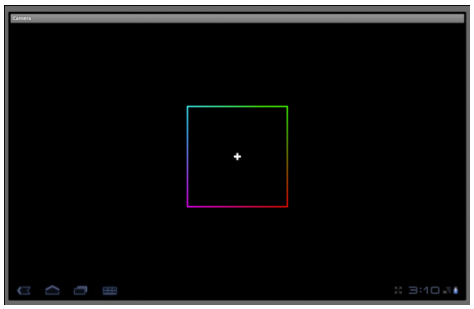


Fig. 4.7.7opengl was in saperated activity in previous design

5. Usage Guide

This chapter will report how to use our application.

5.1 User interface

Fig 5.1.1 User Interface

When the program starts, there is a camera view, a red cross and 6 buttons in the user interface. “Red cross” helps user to aim at a corner. “Capture” button is to start record each corner reading. “Show Distance” button starts the distance measurement function. The result will show on screen left upper corner. “Get Width” starts the length measurement mode.“GEN” button generates the floor plan using the data captured.“Update” button updates the new position on the floor plan.“CLEAR DATA” button clears all the data stored to reset the program.

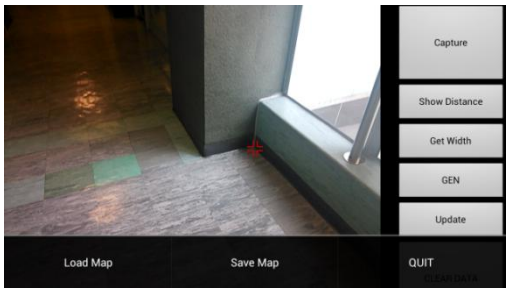


Fig. 5.1.2 Menu bar

There is also menu bar where there are three choices. “Load Map” is used for loading out a pre-generated map. “Save” is used to save the map. “Quit” provides a way to close the program.

5.2 Usage

The steps for using the application is as follows

Generate floor plan

Step 1: Start the application inside a room

Step 2: Point the red cross on the camera view at one corner of the room

Step 3: Press on the capture button to capture one corner.

Step 4: User rotate clockwisely to find the next nearest corner

Step 5: Repeat step 2 to 4 until it reaches the first corner generated

Step 6: Press “gen” button the floor plan can be generated and white cross shows the current standing position

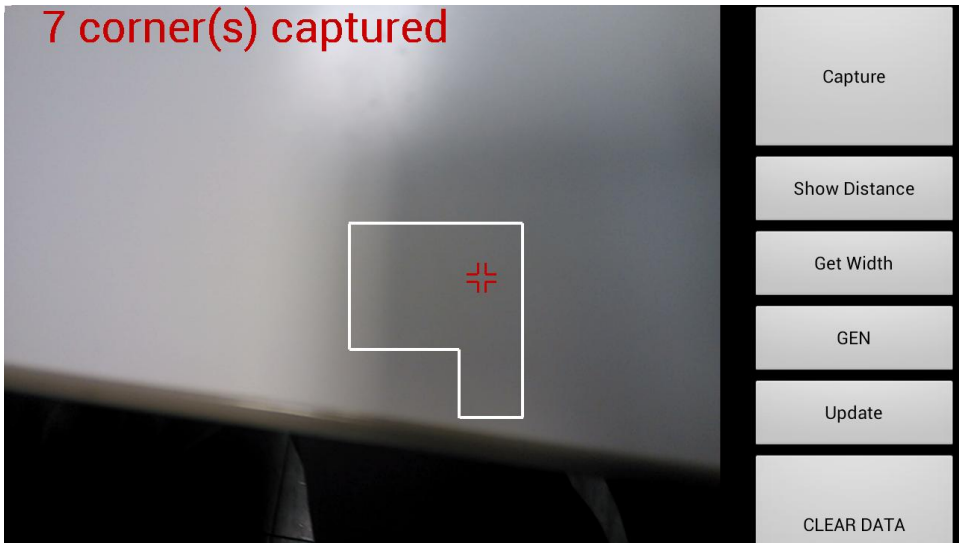


Fig 5.2.1 Floor Plan Generated on screen

Locate on a floor plan (first corner)

Step 1: Start with a floor plan generated on screen

Step 2: Point the red cross on the camera view at the first corner of generation.

Step 3: Press “update” button, the white cross will be updated to show the new position.

Locate on a floor plan (auto mode for 4 corners room)

Step 1: Start with a floor plan generated on screen

Step 2: Point the red cross on the camera view at an arbitrary corner

Step 3: Press “update” button, the white cross will be updated to show the new position

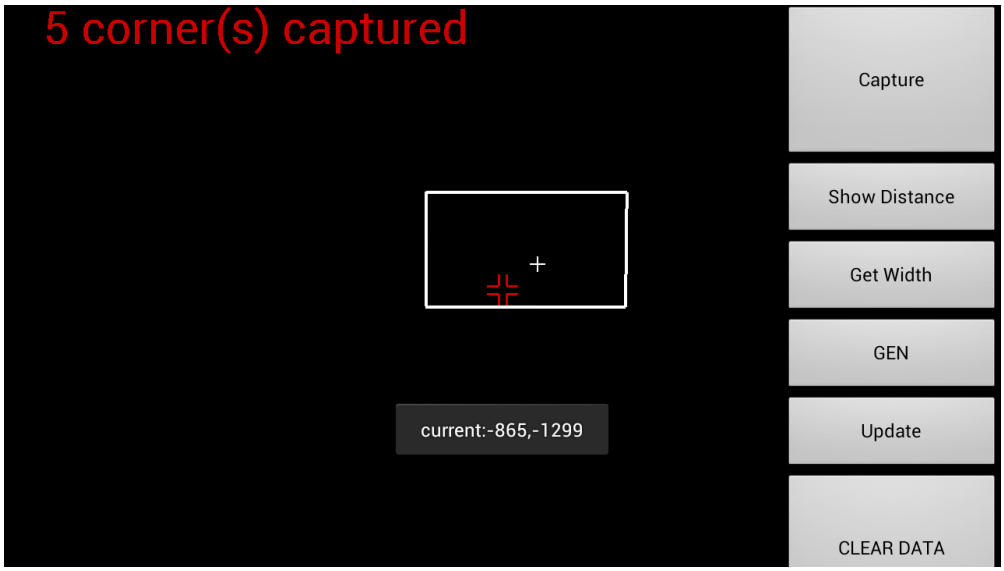


Fig 5.2.2 Location updated

Measure length

Step 1: Press on the “Get width” button to enter the length measurement mode.

Step 2: Aim at one end using the red cross and press the capture button.

Step 3: Aim at the other end and press the capture button. The length measured will be shown on the left upper corner on the screen.

Fig 5.2.3 length measurement result shown on screen

Measure distance

Step 1: Aim at a point using the red cross.

Step 2: Press on the “Show distance” button. The distance from the standing point to the

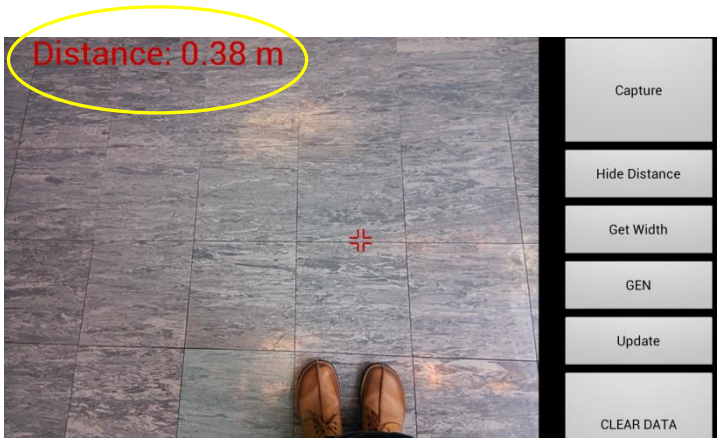


Fig Distance shown on screen

Sve and reload map

Step 1: Click on menu button of the android device.

Step 2: Choose save/load to save or load the floor plan

6. Experiment and testing

6.1 Introduction

In order to test the accuracy of the functions we have implemented, many experiments and testing were performed. In general, we have tested the device orientation angles which are required by other functions as input in the system. We have also tested the measurement function and floor plan generation function.

Note that we also performed a series of similar experiments discussed above in first term. However, we have changed our methodology on obtaining sensor reading. We need to do the experiments again using the new method to check if there is improvement after applying the new method.

Moreover, the changes in device cause us to re-do the experiment performed in semester one so as to maintain a fair experimental comparison. We have changed our development and testing device from Transformer Prime to Xiaomi 2 (detail in chapter 3).

The following sub-chapters will give the details and results of each experiment.
The last sub-chapter will summaries the important conclusion of the experiments.
The experimental data can be found in the appendix.

6.2 Horizontal angle measurement

Introduction

The horizontal angle (azimuth angle) of the device orientation is needed for computing the distance from the device to the corner of a room. We have tried all the combination available in obtaining the device orientation.

Objective

To understand how large the error is when obtaining the horizontal angle of the device using the accelerometer-magnetometer combination, gyroscope, sensor fusion all sensors and just orientation reading respectively.

Setup

Place a protractor on table. Put the device vertical up right to make a 90 degree with the protractor. Capture one reading from the compass and rotate according to the marking on the protractor. Then capture another reading and subtract the two readings can get the required angle. Repeat the experiment with different methods for obtaining the reading.

Top view

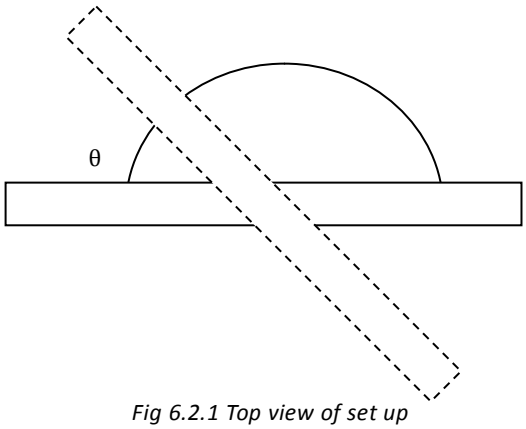
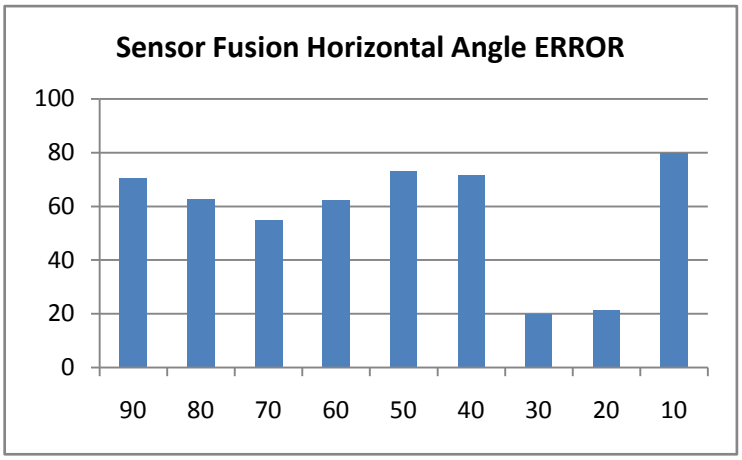
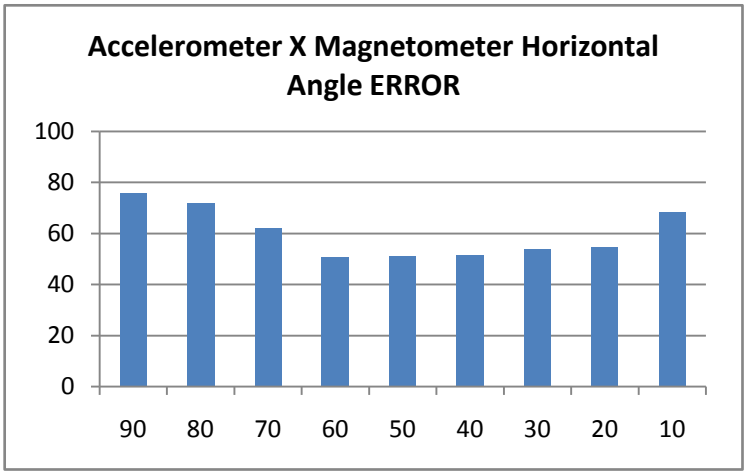
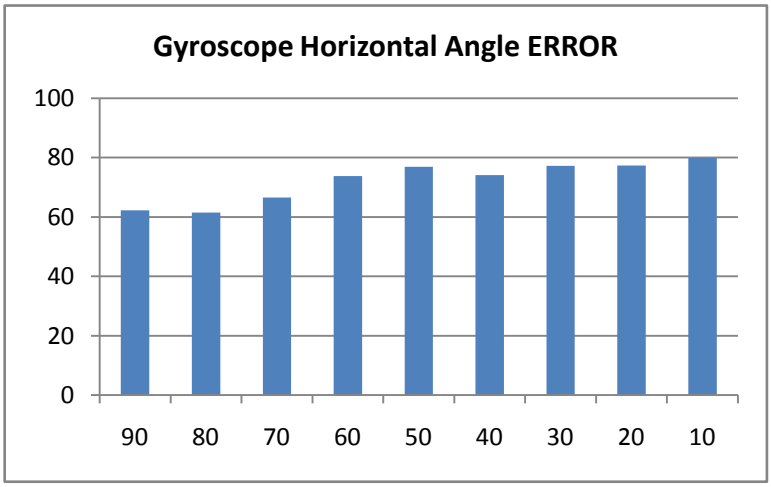


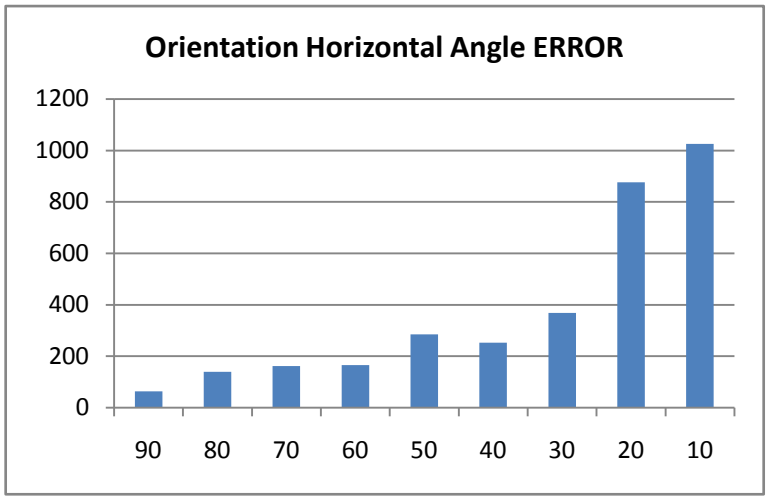
Fig 6.2.1 Top view of set up

Result


Sensor Fusion Horizontal Angle ERROR
100
80
60
40
20
0
90 80 70 60 50 40 30 20 10


Accelerometer X Magnetometer Horizontal Angle ERROR
100
80
60
40
20
0
90 80 70 60 50 40 30 20 10


Gyroscope Horizontal Angle ERROR
100
80
60
40
20
0
90 80 70 60 50 40 30 20 10



Conclusion

The method we used in term one causes error larger than 150%. The error is reduced to about 60%. It can be concluded that Sensor fusion can really improve the inaccuracy of the original method. However the error of 60% is still quite disappointing. The error may propagate and cause error in length measurement in the later stage. Sensor fusion may not be enough for accurate measurement

6.3 Vertical angle measurement

Introduction

The vertical angle (pitch angle) of the device orientation is needed for computing the distance from the device to the corner of a room. We have tried all the combination available in obtaining the device orientation.

Objective

To understand how large the error is when obtaining the vertical angle of the device using the accelerometer-magnetometer combination, gyroscope, sensor fusion all sensors and just orientation reading respectively.

Setup

Measure the pitch angle for step of 10 degree. Use protractor to measure the 10 degree and see the difference from our measurement function and the actual measurement by protractor. Repeat the experiment with different methods for obtaining the reading.

Horizontal view:

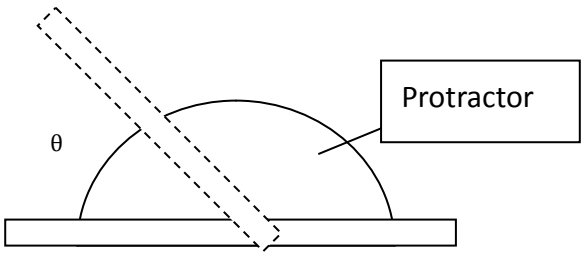
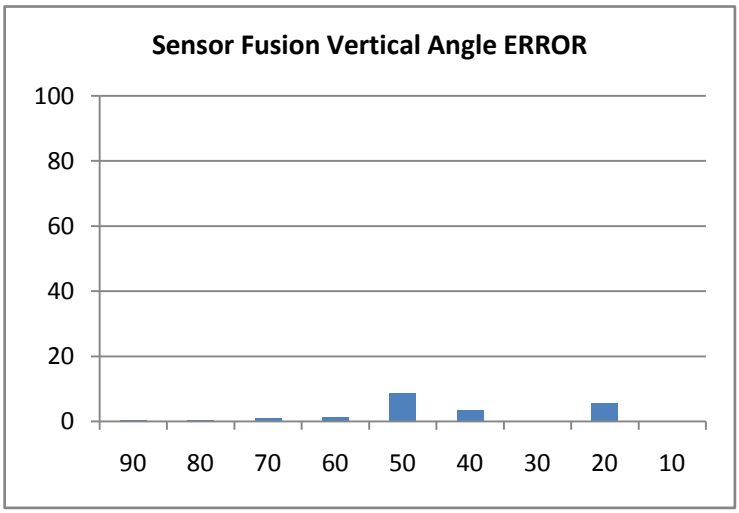
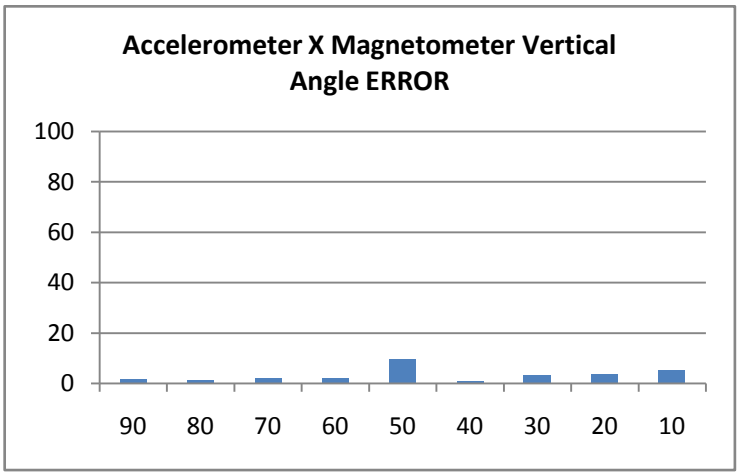
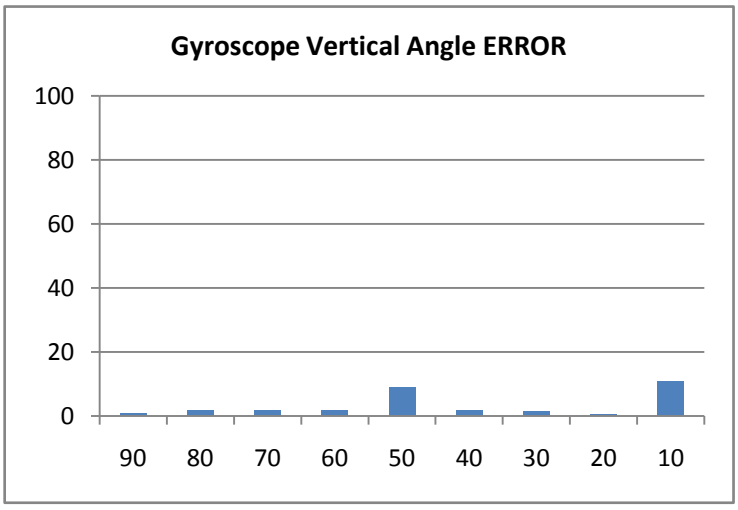


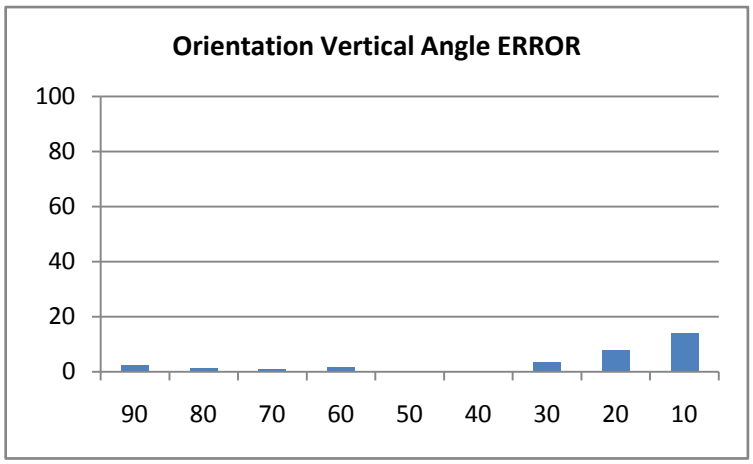
Fig 6.3.1 Horizontal view of setup

Result


Sensor Fusion Vertical Angle ERROR
100
80
60
40
20
0
90
80
70
60
50
40
30
20
10


Accelerometer X Magnetometer Vertical Angle ERROR
100
80
60
40
20
0
90
80
70
60
50
40
30
20
10


Gyroscope Vertical Angle ERROR
100
80
60
40
20
0
90
80
70
60
50
40
30
20
10



Conclusion

Generally for all the four combinations, the error is very small. The error of the vertical angle is accurate enough and could be trusted for distance measurement.

6.4 Distance measurement

Introduction

In our application, distance measurement function is an intermediate step of finding the length between the devices. We need to know whether this step is accurate for us to proceed to the length measurement.

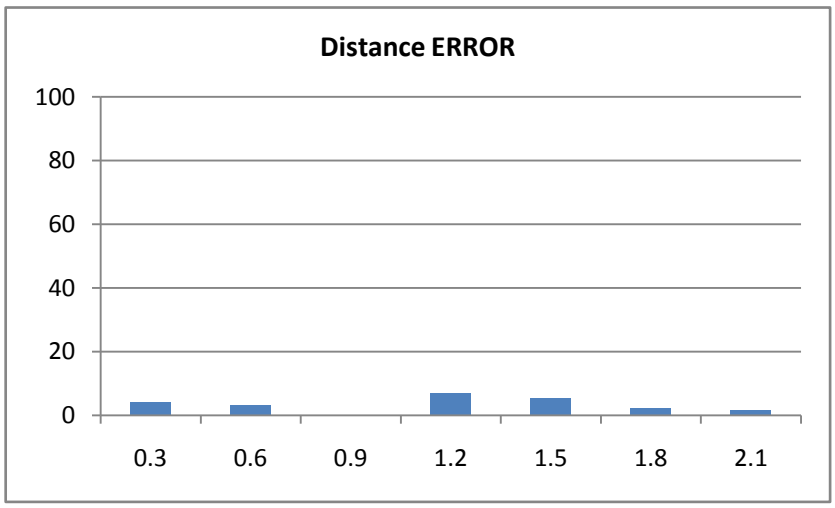
Objective

In this experiment we want to find the error in our distance measurement step.

Setup

Stick a measurement tape on the ground to mark the length. Use our device to point on some markings of the tap to measure the distance. The output of the measurement should meet with the corresponding marking of the tape. Check the difference and find out the error.

Result



Conclusion

The error is quite small and acceptable. Our application is able to measure distance correctly using the distance measurement mode.

6.5 Length measurement

Introduction

Length measurement function between the two corners is one of the major functions of our program. In this part, we do experiments on measuring length between two corners of a room.

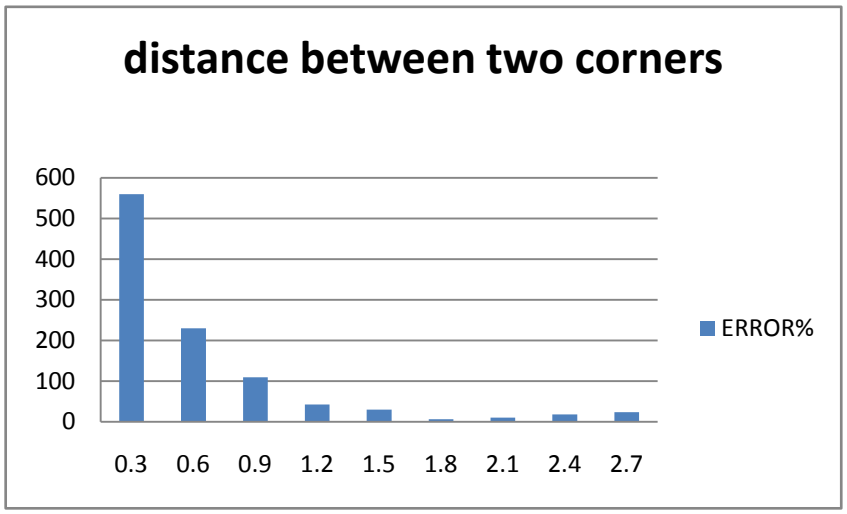
Objective

Find out the error of our length measurement function.

Setup

Point our device with two horizontal points with known length measured by ruler. Then use the length measurement function of our application to measure the length and record the difference from the actually measurement.

Result



Conclusion

The error obtained is quite large. The error is probably come from the error of the horizontal angle. Although Sensor fusion is applied to get a more accurate result, it is not enough to use for measurement. Our application still needs improvement on measurement length between two points.

6.6 Floor Plan Generation

Introduction

Floor plan generation is another major function of our application. In floor plan generation, the aspect ratio of the sides of a floor plan is the major factor determining the accuracy of our function.

Objective

To see whether our application perform floor plan generation function properly.

Setup

We have chosen SHB 102 fun room as our experiment venue. We do testing on floor plan generation inside the room. We first measured the length and width with physical instrument and calculate the ratio. Then we use our application to create a floor plan can measure the aspect ratio and compare with the actual measurement.

Result

Average error after 5 trials: 9.96%

Conclusion

The error of floor plan generated could be acceptable but is not accurate such that it can be used for measuring uses. There are still rooms for improvement. There may be other factors affecting this error since the error should be larger as the horizontal angle obtained for floor plan computation is very large but the result of this experiment is not very large.

6.7 Floor Plan Generation Algorithm

Introduction

Our floor plan generation algorithm can generate a set of points of a floor plan by inputting an array of angle. The most important factor for the accuracy is whether the distance between the first point and last point generated is very small. In this test, we want know whether the two points merges together. This experiment is done in

Objective

To test whether our floor plan generation algorithm can work normally as expected.

Setup

Using command line on PC, we write a simple java program and enter some different degrees of angle input to see distance between the last point and the first point is it small enough. We have entered 10 sets of different angles of square rooms to generate the coordinates of the corner of the floor plan

Result

On average, the minimum distance between the first point and the last point generated = **0.00656** ,

which can be considered as small. Thus, the difference between the first and the last point is quite small that can be ignored when the angles array are appropriately inputted(meeting the assumption that the person is inside the room to capture the corners).

Conclusion

Our floor plan generation algorithm works fine if the input angles array obey the assumption we have made such as the user is standing in the room to capture is scene. It crashes when four corner of the room is in front of the user but this case is impossible if the use is standing inside a room.

6.8 Other testing done

There were some other experiments done in the first semester. We compared our work with some commercial product including smart tools and MagicPlan discussed in chapter one of this report. This was done by doing the similar experiment above using the commercial product. The experimental results showed that the error was similar to our program. We cannot directly compare the result from the first term since we have changed our development and testing device. This would not be a fair test if the results are directly compared.

Due to the resources and time limitation, we have not done the same experiments using the new device again. But it is expected that our application work better since we have increased the accuracy on obtaining sensor reading by sensor fusion method.

6.9 Conclusion and discussion

The vertical (pitch) angle and the distance measurement is accurate and acceptable while the horizontal angle and the length measurement is not accurate. The floor plan generated ratio is also acceptable.

The change in sensors data obtaining methodology had improved the accuracy of these reading. However, the sensor fusion is still not enough to obtain a very accurate horizontal (azimuth) angle which causes error propagation to the length measurement. The error is still large even it was reduced a lot.

Sensors fusion cannot solely solve the very large error. Possible reason contribution to the huge error maybe the device sensors have its own defect. That means the sensors itself is inaccurate. Or electronic devices in the surrounding affecting the magnetometer reading so that reading obtained have a large error. Thus although sensor fusion brought us a more accurate and steady reading, there are other factors contributing to the error we have not figured them out and solved them.

7. Difficulties and limitations

We have faced many difficulties during the project. This chapter will discuss on the limitations and problems we have encountered.

Sensors error

In semester one, we used the Android API SensorManager to obtain and use the device orientation directly. Experiment showed the error of magnetometer is large. The magnetic field sensor was noisy and inaccurate. In semester two, device orientation is obtained by sensors fusion method. Although sensor fusion brought us more steady reading, there are still other factors affecting the accuracy. The error was probably caused by other devices in the room affecting the normal function of the sensors. This is a hardware issues limitation.

Limitations in algorithm

Assumptions have been made in the algorithms designed in this project. In measurement, we have estimated a general height of device from the ground. Although experiment showed only a few percent of error, the height does vary for different users holding the device and would cause inaccurate measurement. In floor plan generation, the corners of the indoor room are assumed to be of ninety degree. There are many indoor rooms with different angles and our application is not suitable for those rooms. In localization, we required the user to capture only the first corners. Or any corners for 4 corners room. This restriction may cause a less comfortable user experience. We tried to reduce the inconvenience by providing assistance for user to locate the first corner.

Studied on 3D aligning

We have tried to use a openGL 3D library JPCT to add a 3D world into the user interface layout. We intended to align the real world device camera with 3D virtual world camera. However, many problems occurred such as the 3D world cover all the screen and the camera “jump” on moving. We didn’t have enough time to solve all the problems since we were not familiar with the 3D development platform of JPCT. Sadly, we decided to give up the addition of 3D opengl world to utilize time on other part.

Others

There were minors difficulties we faced and got used to. It was at first difficult to adapt to android development. Research materials and papers were hard to understand. Computing resources are still limited on mobile devices. We still needed to handle memory usage in careful way.

8. Conclusion

8.1 Summaries

To conclude this project, we have first look into the current issues of mobile devices and studied application and software on the market to design what are we doing. Then we did some review on the research of others on the related research topic. After that, we start to design and implement the project work. While we are implementing the project, we did some testing and experiments to see the error of functions the program performs. We analyzed the error and proposed possible reasons. We have also tried to improve the accuracy by changing the methodology of obtaining reading from sensors.

In our system implementation, we first got used to android environment before developing the program. T**he major functions we have implemented included device orientation, measurement, floor plan generation, localization and user interface implementation.** And the device orientation had been changed to sensor fusion in semester two. **Minor functions are restructuring to use the some of the side product like length and distance measurement, database storage.** A minor point is that we have also tried to map the camera to opengl camera in order to device camera world but faced many problems when handling 3D opengl view so this could not be finished.

There are still difficulties in the project like the sensors reading were affected environment. More computers and electronic devices around the device caused the reading from sensors not reliable.

List of items completed in term 1:

- Obtain sensor reading from Android API
- **Distance measurement**
- **Length measurement** with error revealed
- **Floor Plan Generation**
- Experiments and testing

List of items completed in term 2:

- **Change the sensor reading obtaining methodology to sensor fusion.**
- **Localization: updating current position on floor plan.**
- Packaged the distance and length measurement functions as side product of the system
- User interface changed
- Provide database storage and loading of floor plan
- Experiments and testing
- Studied on 3D but cannot finish

8.2 Future work

We have built the measurement, floor plan generation and updating position function in our program. This is the core of the project. On top of this, we can utilize this to build different useful applications. For example, the openGL view can be extended to 3D world for AR gaming application. To achieve this, the opengl camera of the game world has to be aligned with the device camera looking at the real world. This could be done by a set of trials and calibration which are very time-consuming stuff. The floor plan generated can be used as the map of the room. Another example is a3D room design application. The idea is to construct a 3D room and put different object into the room. This can help on indoor design or commercial selling furniture. But the accuracy of the functions may not be suitable for measurement usage.

Since our functions provides many possibilities of using it, a future extension of this project is to restructure the code and wrap the different functions into library and build a documentation for other develop to unitize and create their own applications in their mind.

The functions of floor plan generation can be extended. First the floor plan generation algorithm can be developed to support to 3D map construction, i.e, to extend our current 2D coordinate system to 3D coordinate system to store all the corners of the room. Second, the algorithm can be extended to support the different angles of corners since it currently only support the angle of ninety degree. This requires the change of our algorithm for more flexibility.

The localization algorithm can also be enhanced to support automatic updating or less restricted capture of corners. Since the current method for updating the position on floor plan require user to capture only the first corner or there is only 4 corners in the floor plan. An automatic updating position process can have better user experience. But this is not an easy task may require more time on researching this issues.

Also, the accuracy and reliability of the functions can be further enhanced. Although we have modified to use sensor fusion to obtain the orientation of device, there may be more ways to enhance the accuracy which require more time to investigate.

9. Division of labour

We divided our work into pieces to separate each other work in order to ensure a more efficient progress of project. Most design we need to sit together to discuss so that both of us have a better picture of the system. When anyone of us encountered problem on the individual work, we would raised them out to discuss the solution.

I was focusing on the issues of Android API and other library usage such as the opengl es. I had tried to improve the accuracy of our work by using the sensor fusion technique. Some of the experiments and data analyzes was my task. I had also worked on the database storage of the system.

The following table shows a more detailed division of work:

Parts	Person: Huen Shiu Fung (Steve) Tsang Sze Hong (Felix)
Project initiation	-
Idea fetching	Steve, Felix
Background research	Steve, Felix
Review on Relevant topic	-
Review on Length measurement	Steve
Review on Rendering Indoor Scene	Steve
Review on AR	Felix
Review on indoor positioning	Felix
Review on Sensor Fusion	Steve
Design and implementation	-
Android Activity	Felix
User interface	Felix

Measurement	Steve
Sensor fusion	Felix
Floor Plan Generation	Steve
Localization	Steve
Device orientation	Felix
Data Storage	Steve
Data communication and System Structure	Steve, Felix
Database Storage	Felix
Experiment and testing	-
Device orientation	Steve, Felix
Measurement	Steve, Felix
Floor Plan Algorithm	Steve
Comparison with products	Steve, Felix
Localization	Steve, Felix
Sensor fusion	Steve, Felix

Since most of the experiments have to be done efficiently with two people, it was not easy to separate the work.

10. Reflection

Throughout the development of the project, I have learnt a lot from our supervisor Prof. Lyu and Mr Yau from VIEW lab.

I felt it challenging on developing this project. I have looked through many materials including research publication and other online tutorial on different related topic of our project. Most of them were hard to understand. The project scale is rather large and many difficulties had to be solved during the development. Both soft and hard skills have been trained from this project.

Last but not least, my partner Huen Shiu Fung(Steve) also helped me a lot on giving me different support. I am pleased to have a cooperative final year project partner and worked with him nicely in this year.

11. Acknowledgement

We would like thank our supervisor Prof. Michael Lyu sincerely for his invaluable advice and support. Prof. Lyu arranges an 1-hour meeting on weekly basis for us from his busy schedule to keep our progress up. Throughout the project development, he gave us useful advice, guided our project toward the right direction and kept the track of our work.

Moreover, we would like to express our thanks to Mr. Edward Yau in VIEW lab for giving us useful technical support and providing us insights and suggestions to solve the difficulties we have encountered during the project development time.

12. References

[1]Paul A. Zandbergen and Sean J. Barbeau (2011). *Positional Accuracy of Assisted GPS Data from HighSensitivityGPSenabled Mobile Phones.*Journal of Navigation, 64, pp 381399 doi:10.1017/ S0373463311000051

[2]Günther Retscher (2007). *Test and Integration of Location Sensors for a Multisensor Personal Navigator*. Journal of Navigation, 60, pp 107117 doi:10.1017/S037346330700402X

[3]Adutya Sankar(2012) *Capturing Indoor Scene with Smartphones* 2012 ACM 978-1-4503-1580-7/12/10

[4]A. Criminisi(1999) *Single View Metrology* University of Oxford Oxford, UK, OX1 3PJ

[5]. Google maps 6.0
http://googleblog.blogspot.hk/2011/11/new-frontier-for-google-maps-mapping.html
Access on 25/11/2012

[6]. Nokia Destination maps
http://conversations.nokia.com/2012/07/16/nokia-leads-the-way-with-indoor-mapping/
Access on 25/11/2012

[7]Android API
Developer http://developer.android.com/reference/packages.html
Access on 23/10/2012

[8.] Lauren Darcey and Shane Conder *Augmented Reality GettingS Start On Android.* http://mobile.tutsplus.com/tutorials/android/android_augmented-reality/

[9.] Lauren Darcey and Shane Conder*Android Barometer Logger: Acquiring Sensor Data* http://mobile.tutsplus.com/tutorials/android/android-barometer-logger-acquiring-sensor-data/

[10] "openGL ES" http://www.khronos.org/opengles/

[11] Ng Ka Hung, Chan Hing Fat (2011) *Digital Interactive Game Interface Table Apps for ipad*

[12]Guanghui Wang (2005) *Single view metrology from scene constraints* Image and Vision Computing 23 (2005) 831–840

[13]Google android documentation for gyroscope http://developer.android.com/reference/android/hardware/SensorEvent.html#values

[14]Sensor fusion

http://www.thousand-thoughts.com/2012/03/android-sensor-fusion-tutorial/1/

[15]OpenGL tutorial by

http://fecbob.pixnet.net/blog/post/34682793-android-opengl-es-draw line-

[16]3D opengl on Android: JPCT

http://www.jpct.net/

[17] Wifi indoor positioning from Monoj-Jumar-Raja web

https://sites.google.com/site/monojkumarraja/academic-projects/wi-fi-indoor-positioning-system

[18] Smart tools

https://play.google.com/store/apps/details?id=kr.aboy.tools&feature=search_result#?t=W251bGwsMSwxLDEsImtyLmFib3kudG9vbHMiXQ..

Appendix

Development environment

Type	Software	Version
IDE	Eclipse	3.7.1.M20110909-1335
Software	Android DDMS	18.0.0.v201203301601-306762
Software	Android Development Tools	18.0.0.v201203301601-306762
Software	Android Hierarchy Viewer	18.0.0.v201203301601-306762
Software	Android Traceview	18.0.0.v201203301601-306762
Software	Eclipse XML Editors and Tools	1.1.201.v201108151912
Software	NVIDIA Debug Manager for Android NDK	14.0.1.201202211602
Software	Structured Source Editor	1.3.1.v201108191312
Software	Structured Source Model	1.1.601.v201108151912
SDK	Android SDK	2.2
SDK	Android SDK	2.3.1
SDK	Android SDK	2.3.3
SDK	Android SDK	3.0

SDK	Android SDK	3.2
SDK	Android SDK	4.0
SDK	Android SDK	4.03

Experimental Data

6.2 Horizontal angle

Sensor Fusion										
Actual	Record1	Record2	Record3	Record4	Record5	Record6	Record7	Record8	Record9	Record10
90	36.0389423	29.3541393	14.6669092	30.8200054	26.4901772	28.5179138	26.1858635	26.6287193	20.7494335	25.0437737
80	41.8389301	31.5794468	30.3817978	31.5050144	31.465559	29.7997494	27.9637566	21.3937645	25.1453066	27.6725969
70	30.0116129	27.4018106	28.8800879	30.9095039	24.0060043	21.0653696	20.6999655	26.3891468	83.323514	23.2730322
60	27.1531687	22.1155882	25.5571203	18.3310852	41.7058525	13.9329987	15.4226098	19.7563505	20.0791183	22.3086352
50	21.8473511	15.1552124	14.099884	12.5166931	13.9933472	12.6984253	16.133606	12.877594	14.3336487	
40	10.6247864	11.8028564	13.2588806	13.2278137	11.0191956	9.40921021	10.0176697	10.2949524	10.9382935	13.6059875
30	26.3310852	24.7658691	24.5379028	23.4838867	23.5956116	23.8653564	26.3123779	23.9883728	20.7183228	22.3538818
20	16.1549988	15.6371765	16.583374	15.3536377	15.2488098	16.5079346	15.6980896	14.6890869	14.6227112	16.3811035
10	2.64746094	2.84744263	2.74057007	1.97616577	2.10699463	1.82107544	1.90142822	1.1171875	0.73861694	2.15597534

Sensor.GYRO										
90	42.3117905	35.5246964	35.6342793	38.0256844	34.2081871	32.2043915	34.6384926	27.2284145	28.1451645	31.7006931
80	29.0332985	29.2751122	25.8340549	30.3738461	29.004179	35.3234787	33.9892788	27.4672489	35.8573208	32.2437439
70	24.5421543	20.018239	20.4971132	16.904911	45.9573288	10.1768645	20.6852798	22.9425254	27.8121538	24.896863
60	19.3075924	18.2144418	16.2920146	26.6537104	11.360136	13.4156928	13.0252523	12.9264722	13.8421087	12.0312715
50	20.9370751	10.5779002	9.36973521	11.4809567	9.51902986	13.9580336	9.48629832	8.33728027	8.79309082	12.8722908
40	13.7469177	12.5872498	13.4155884	8.52310181	8.41586304	11.4062805	8.86672974	7.81781006	10.6183777	8.21682739
30	11.1498718	8.82086182	7.3885498	0.92871094	7.73605347	9.03445435	6.71572876	4.62634277	6.26797485	5.76397705
20	7.11618042	4.80355835	4.5635376	5.43215942	4.12130737	4.90460205	4.89608765	3.8298645	3.18917847	2.49584961
10	2.64746094	2.84744263	2.74057007	1.97616577	2.10699463	1.82107544	1.90142822	1.1171875	0.73861694	2.15597534

Sensor.acce_mag										
90	20.7562561	22.5444031	22.2932434	21.1940613	23.5897217	21.0934448	20.7466125	20.157196	22.5293274	24.03302
80	17.9855042	23.3153381	22.0525818	26.4317322	25.6878967	21.2402039	24.8351746	22.8475342	23.1845703	17.9855042
70	21.928772	25.665863	27.4101868	26.362793	24.8576355	25.6872559	31.2518616	27.305542	28.2704773	25.660614
60	24.6047058	30.2756348	30.3727722	27.6678467	30.1427612	25.5620117	32.4280701	33.1477966	29.5686646	33.1229248
50	22.3431396	28.0945129	28.2857666	18.8095703	27.0513	32.6890259	30.6377563	30.2448425	26.0009766	
40	16.1650696	19.0703735	19.7976074	19.1392212	16.1234741	10.500885	24.377655	26.848938	24.9660645	16.1650696
30	11.4405518	13.9674683	15.9285583	12.9249268	13.1197815	11.1539917	13.994873	17.6502686	13.6903381	14.4075012
20	6.8237915	7.79391479	9.98577881	9.24798584	9.02282715	9.46130371	9.37161255	8.41412354	10.0645142	10.115509
10	3.46942139	2.16009521	2.87741089	3.34136963	3.83709717	4.11322021	0.50167847	3.74533081	3.90887451	3.46942139

Sensor.TYPE_ORIENTATION										
90	54.4031677	208.365356	198.023071	201.795067	77.2659149	65.8927917	71.3404083	67.1193695	198.151772	324.121925
80	179.877617	184.524902	323.943695	190.781059	54.2151489	190.305672	208.034149	186.550659	199.137703	195.15966
70	36.0874023	193.468773	322.207405	188.832939	58.0202332	319.899132	193.201416	7.3407135	324.952049	188.906395
60	209.070847	144.682739	189.218277	55.4406128	188.136948	185.715111	182.5597	327.13195	54.819046	54.2558899
50	329.795853	187.464981	184.787285	328.958801	187.278252	280.297699	49.1287384	187.747749	189.944122	
40	189.687988	36.1312103	37.011261	337.257767	173.991425	337.905754	34.8193665	38.3195038	190.213303	37.5549011
30	174.606026	26.9848175	173.002762	171.271271	264.454086	343.297775	27.4366608	173.936676	24.8752136	24.2824249
20	202.615807	353.410812	15.462326	165.061295	16.5396576	205.662498	353.340263	353.06955	262.921555	23.4751587
10	4.15959167	7.01472473	211.219795	1.25328827	8.52555847	359.130981	6.26670837	8.47462463	359.30069	159.873932

6.3 Vertical

Sensor Fusion										
Actual	Record1	Record2	Record3	Record4	Record5	Record6	Record7	Record8	Record9	Record10
90	-91.28243	-91.60553	-91.62591	-90.07362	-90.04022	-90.03405	-89.980515	-89.925934	-89.924736	-89.93152
80	-79.66784	-79.79164	-79.81167	-79.811005	-79.790146	-79.76234	-79.46991	-79.366104	-79.35057	-79.32771
70	-69.32699	-69.53224	-69.51033	-69.43591	-69.39788	-69.33304	-69.31161	-69.24151	-69.19115	-69.21657
60	-58.487465	-59.09771	-59.28335	-59.25706	-59.2425	-59.302944	-59.26212	-59.211777	-59.17955	-59.211876
50	-50.54806	-50.68692	-50.617798	-50.64782	-50.640724	-50.679893	-50.672466	-50.65781	-50.585793	
40	-38.717613	-38.65347	-38.52865	-38.327488	-38.275932	-38.619495	-38.65794	-38.708595	-38.70551	-38.71779

30	-28.03636	-29.817957	-30.12744	-30.161552	-30.181267	-30.20457	-30.231785	-30.255337	-30.30103	-30.339699
20	-19.07501	-18.993412	-18.954666	-18.937664	-18.890257	-18.8052	-18.78505	-18.775131	-18.763733	-18.670706
10	-9.788406	-9.888117	-10.013824	-9.996859	-10.065063	-10.063005	-10.056862	-10.033341	-10.108573	-10.104712

TYPE.GYROSCOPE										
90	-91.07477	-90.57232	-90.5433	-90.560715	-90.57396	-90.83956	-90.89083	-90.868546	-90.84388	-90.84422
80	-80.98563	-81.52533	-81.71706	-81.74264	-81.322624	-81.328354	-81.25305	-81.27946	-81.290764	-81.2691
70	-70.869934	-71.20732	-71.33283	-71.27209	-71.29534	-71.22004	-71.27636	-71.26566	-71.26573	-71.25605
60	-61.238693	-61.20455	-61.025337	-60.988525	-60.960564	-60.990673	-60.938004	-60.95008	-61.05374	-60.96266
50	-50.54806	-50.68692	-50.617798	-50.64782	-50.640724	-50.679893	-50.672466	-50.65781	-50.585793	
40	-39.58989	-40.033752	-40.671608	-40.722614	-40.780132	-40.821075	-40.866245	-40.90408	-40.896317	-41.07734
30	-30.012102	-30.256874	-30.308245	-30.289465	-30.459057	-30.534313	-30.552896	-30.610971	-30.628996	-30.63099
20	-19.737371	-21.217321	-20.918026	-21.097183	-21.099995	-21.098581	-17.948208	-18.615602	-19.306046	-19.74434
10	-10.167775	-11.199753	-11.244342	-11.28499	-11.300948	-11.28389	-11.265695	-11.123879	-11.033056	-10.931587

ACCE+MAG										
90	-91.57044	-91.66653	-92.01987	-91.79519	-91.4581	-91.6763	-91.567375	-91.567375	-91.46385	-91.79168
80	-80.95572	-80.69206	-80.78255	-80.469215	-81.31867	-81.42966	-81.00358	-81.10361	-80.91519	-81.22488
70	-71.29606	-71.65135	-71.19006	-71.90016	-71.79311	-71.614944	-71.90016	-71.43713	-71.43713	-71.579636
60	-60.771824	-61.48566	-61.110416	-61.38641	-61.124012	-61.03541	-61.38641	-60.98072	-61.6769	-61.277924
50	-49.64368	-50.50418	-50.299896	-50.35933	-50.431892	-50.632442	-48.081715	-50.679825	-50.416718	

40	-39.45743	-40.765705	-40.765705	-39.83272	-40.679363	-40.431812	-40.310547	-40.21071	-40.44463	-40.44463
30	-30.661503	-31.048819	-31.010338	-30.700924	-31.068577	-31.087118	-31.068577	-30.894411	-31.320591	-31.165302
20	-19.423029	-19.175455	-19.423029	-19.50575	-18.9825	-19.537146	-19.391306	-19.245789	-18.744043	-19.561243
10	-9.289554	-9.631609	-10.050872	-10.15276	-10.993056	-10.775469	-11.182379	-11.275094	-10.835932	-10.980794

TYPE.ORIENTATION										
90	-86.770477	-87.137279	-87.342806	-87.314932	-87.345825	-87.312111	-91.233302	-86.48085	-92.596513	-86.527125
80	-80.95466	-80.928446	-80.924416	-80.842078	-80.868764	-80.898871	-80.957392	-80.986038	-81.20776	-80.897102
70	-69.132251	-69.157953	-69.16712	-69.338947	-69.052106	-69.579424	-69.274176	-69.539932	-69.275699	-69.132251
60	-61.233151	-60.974116	-60.970223	-61.035411	-60.886191	-61.048258	-60.991609	-60.991991	-61.024844	-61.233151
50	-50.092525	-50.011642	-49.989209	-49.951366	-49.962188	-50.462503	-49.984154	-49.998641	-49.571262	-49.671213
40	-40.174596	-40.18132	-39.966271	-40.018061	-40.010647	-39.803296	-39.935734	-39.876246	-39.84939	-39.620182
30	-30.97251	-30.960223	-31.030031	-31.041174	-31.019707	-31.034129	-30.835306	-31.030028	-31.060391	-31.165299
20	-21.168346	-21.070112	-20.951153	-21.525191	-21.434547	-21.601836	-21.650542	-21.833145	-22.205288	-22.101635
10	-11.125434	-10.940728	-11.176831	-11.230604	-11.221768	-11.284535	-11.484219	-11.583233	-12.101952	-11.845962

6.4 Distance

Distance					
0.3	0.33	0.32	0.3	0.28	0.33
0.6	0.66	0.62	0.64	0.57	0.61
0.9	0.94	0.92	0.87	0.87	0.9

1.2	1.21	1.4	1.6	1.08	1.12
1.5	1.7	1.71	1.8	1.3	1.4
1.8	1.83	1.86	1.72	1.67	1.72
2.1	2.31	2.12	1.97	2.08	2.2

6.5 Length

Actual	Record1	Record2	Record3	Record4	Record5
0.3	0.11	0.09	0.07	0.07	0.12
0.6	0.24	0.32	0.49	0.33	0.19
0.9	0.44	0.37	0.37	0.49	0.36
1.2	0.54	0.56	0.55	0.74	0.42
1.5	1.21	0.72	0.81	0.83	0.84
1.8	1.83	1.37	1.18	0.97	1.42
2.1	3.4	1.19	1.18	1.13	1.22

6.6 Floor Plan

Actual	Width	Length	Width/Length
	480	760	0.631578947
Measured	1.5	2.2	0.681818182
	1.7	2.4	0.708333333

	1.6	2.4	0.666666667
	1.3	2.3	0.565217391
	1.2	2.2	0.545454545