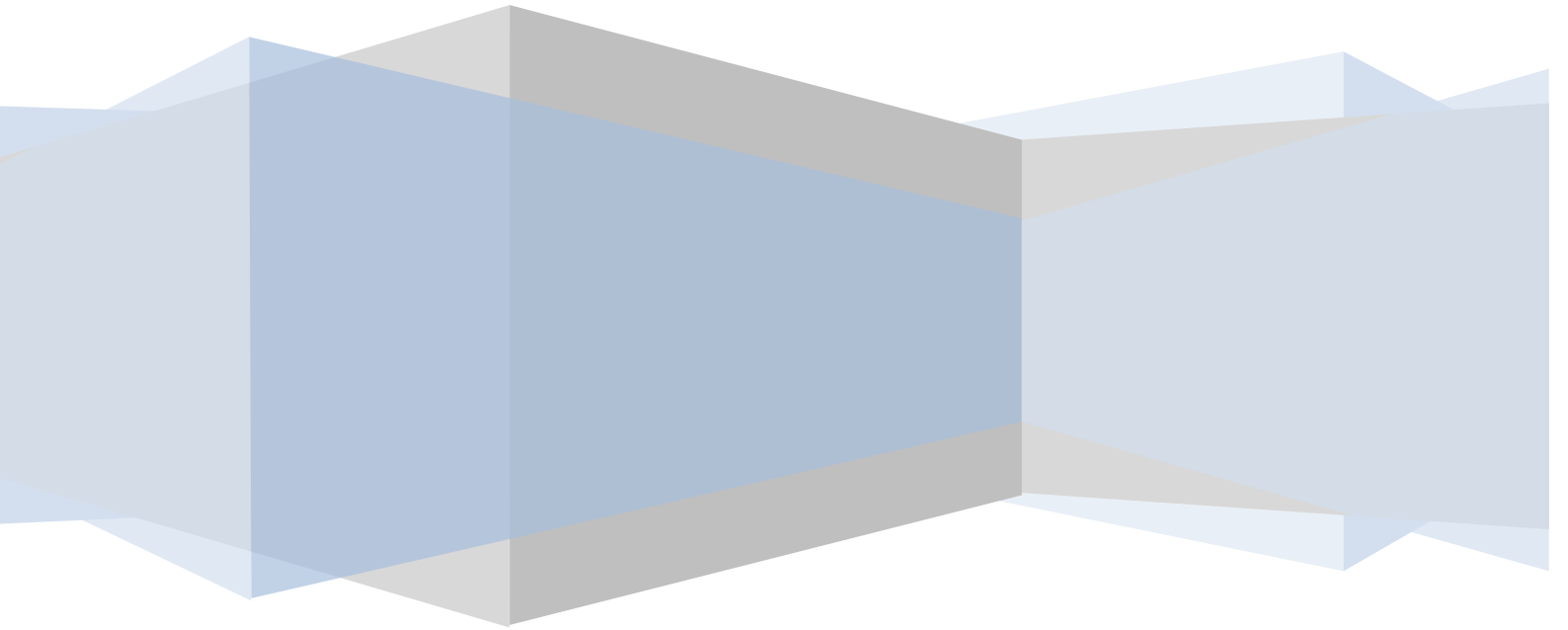
Masarykova Univerzita

Localization based on smartphone sensors

Activity report

Thomas Fardeau



Abstract

This activity report is describing the work that has been done to answer a problem given by Ahmad ABBADI. ABBADI needed to localize a robot with precision using only a smartphone.

The chosen solution is to build an android application using both sensors data and GPS data lo localize the robot continuously.

To complete this application and be even more precise, a video feed will be provided by the application and could later be used in some visual odometry algorithm.

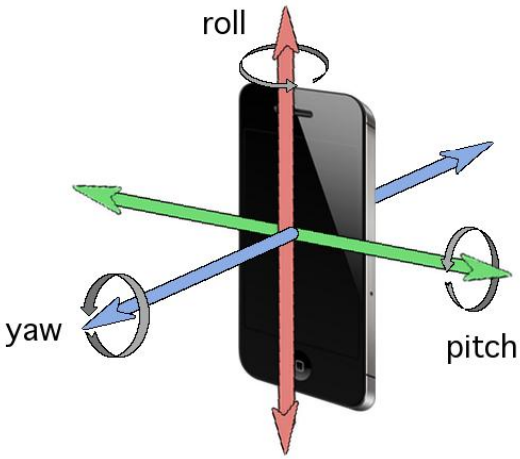
Table of contents

1- Introduction

1.1 The accelerometer

An accelerometer is an electromechanical device that will measure acceleration forces. These forces may be static, like the constant force of gravity pulling at your feet, or they could be dynamic - caused by moving or vibrating the accelerometer.

There are many different ways to make an accelerometer. Some accelerometers contain microscopic crystal structures that get stressed by accelerative forces, which causes a voltage to be generated. Another way to do it is by sensing changes in capacitance. If you have two microstructures next to each other, they have a certain capacitance between them. If an accelerative force moves one of the structures, then the capacitance will change. Add some circuitry to convert from capacitance to voltage, and you will get an accelerometer.



1. Three axis accelerometer in a smartphone

1.2 The magnetometer

The magnetometer sensor in your tablet or smartphone utilizes the modern solid state technology to create a miniature Hall-effect sensor that detects the Earth's magnetic field along three perpendicular axes X, Y and Z. The Hall-effect sensor produces voltage which is proportional to the strength and polarity of the magnetic field along the axis each sensor is directed. The sensed voltage is converted to digital signal representing the magnetic field intensity.

In addition to general rotational information, the magnetometer is crucial for detecting the relative orientation of the device relative to the Earth's magnetic north.

1.3 The gyroscope

Generally, a gyroscope is a device for measuring or maintaining orientation, based on the principles of conservation of angular momentum. A conventional (mechanical) gyroscope consists of a spinning wheel mounted on two gimbals which allow it to rotate in all three axes. An effect of the conservation of angular momentum is that the spinning wheel will resist changes in orientation. Hence when a mechanical gyroscope is subjected to a rotation, the wheel will remain at a constant global orientation and the angles between adjacent gimbals will change.

An important note to be made is that whereas the accelerometer and the magnetometer measure acceleration and angle relative to the Earth, gyroscope measures angular velocity relative to the body.

1.4 IMU (Inertial Measurement Unit)

An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers.

1.5 Android

The solution to the problem given as an internship subject can be implemented on an Android smartphone.

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for smartphones and tablets. Android is offering a lot of ways to control every one of the smartphone sensors but also easy solutions to get GPS data and video feeds.

The language used to code Android applications is Java.

2- Activity report

2.1 The problem to solve

The internship subject is "localization based on smartphone sensors" and it is asking the question: how to localize a robot using a smartphone.

Nowadays, there is a lot of existing technologies that can localize a device and the most well-known one is GPS. However, this technology may be insufficient is some particular cases. For example, when the device is situated in some building or going through a tunnel, the GPS localization may be lost.

This is why we need to use a more precise way to localize the robot. The solution is to build an inertial measurement unit thanks to the sensors provided by the smartphone.

2.2 The solution

To localize the robot, we need different kind of information:

- Sensors data, required to build an inertial measurement unit. This unit will give the precise orientation of the device.

- GPS data, required to localize the device with its geographical coordinates (latitude, longitude and altitude).

In order to have several ways of localizing the robot, we also want to get a video feed that will be later used in a visual odometry algorithm.

To get all of those needed data, I had to create an android application.

2.2.1 Sensors data

From all the sensors available in a smartphone, the application is going to use only three: the accelerometer, the magnetometer and the gyroscope.

Each of those sensors will give some indications on the device's orientation and will bring a complement of information to the GPS data.

To use the sensors of a smartphone in an application, there are some existing tools such as SensorManager or SensorEventListener.

These tools are used to tell the smartphone which sensors are watched by the app but also how we want to get the data from those sensors. More specifically,

the SensorManager is used to register the wanted sensors, and the SensorEventListener are receiving pieces of information from the sensors such as the raw data that is measured or the availability of the sensor.

The sensorManager can also be used to change the acquisition rate of the sensors data and the application is exploiting this possibility through a setup activity.

2.2.2 GPS data

The GPS data is also needed in this application to ensure a precise localization of the device when the GPS data is available. When the GPS is not available, the localization is ensured by the IMU. But using both GPS and IMU is providing a much more precise localization of the device.

Some authorization is needed to use the GPS localization in an android application. In this application, the authorization is granted in the code and is not asked to the user. But in the future, Android will be forcing its developers to ask the application's user for this kind of authorization.

The GPS data is easy to get using the existing Android tools. In fact there are two tools dedicated to get the GPS data is the LocationManager and the LocationListener. With the LocationManager you can register the way to listen to GPS data. That means you can set both the acquisition rate but also the source of the data (device GPS or network localization).

As the acquisition rate can be set by the LocationManager, the application is designed in such a way that this rate is modifiable in the setup activity.

2.2.3 Video feed

In this application, a video is also taken in order to have a third way of localizing the device. This video will be use in the next steps of this project in a visual odometry algorithm.

To take a video in an Android application, there are several parameters to set. A preview of what the camera is filming is compulsory but on this application, seeing what the camera is filming is not important to the user. This is why I made the choice to hide the preview from the user by reducing it to the smallest size possible.

In the process of setting up the preview, there is a nedd to choose a size for the preview and, of course, we need to choose the best size so that the video does

not look distorted. In this application, the best choice possible for each phone will always be chosen because the application is listing all sizes possible and choosing the best resolution of all.

There are also a lot of specific parameters that can be set to have a better video such as fps (frame per second) rate of the video. This aspect of the video can be changed by the user in the setup activity.

This application was developed following a simple solution to transfer the video file to the server. The first step is to record the video and save it in the device. Once the user decides to stop collecting data, the recording stops and the video is sent to the server all at once. After this step, the video file is erased from the device and the application gets back to the home activity.

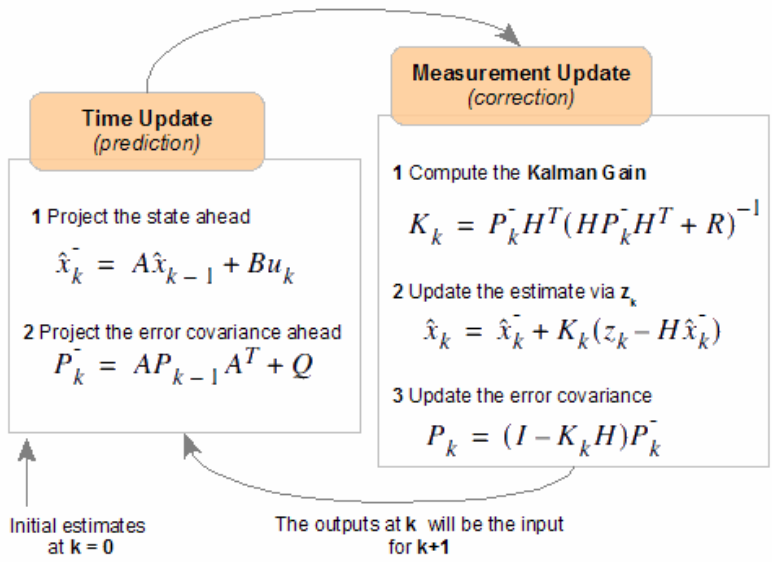
2.2.4 Sensor fusion

The common way to get the attitude of an Android device is to use the SensorManager.getOrientation() method. This method is based on the accelerometer and magnetometer output. The acceletometer provides the gravity vector and the magnetometer works as a compass. The information from both sensors suffices to calculate the device's orientation. However both sensor outputs are inaccurate and include a lot of noise.

The gyroscope in the device is far more accurate and has a very short response time. Its downside is the dreaded gyro drift. The gyro provides the angular rotation speeds for all three axes. To get the actual orientation those speed values need to be integrated over time. This is done by multiplying the angular speeds with the time interval between the last and the current sensor output. This yields a rotation increment. The sum of all rotation increments yields the absolute orientation of the device. During this process small errors are introduced in each one of these iterations. These small errors add up over time resulting in a constant slow rotation of the calculated orientation, the gyro drift.

In order to have a precise data of the orientation of the phone and its position's evolution we need to fuse the data of those different sensors. This fusion will compensate for each of the sensor's disadvantages.

This fusion is made by using a Kalman filter. This filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance when the condition of a linear Gaussian system is met.



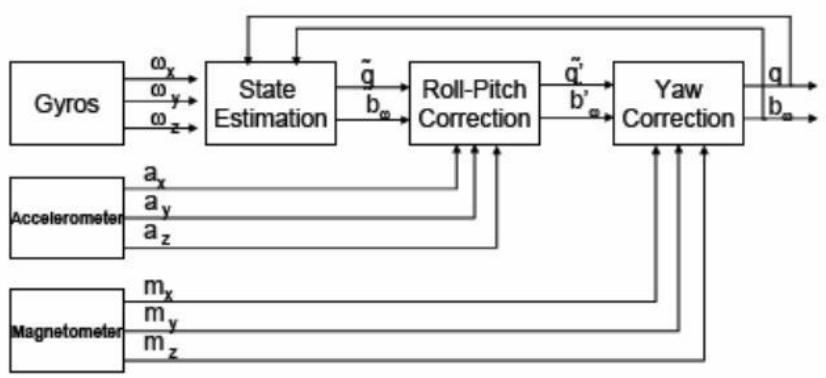
2. Schematics of how a Kalman filter works

In these equations:

- x_k is the estimate of the signal at the time k.
- P_k is the error covariance at the time k.
- K_k is the Kalman gain at the time k.
- A,B and H are matrixes coming from the modeling of our situation.
- $\hat{x}\bar{k}$ is the prior estimate representing the rough estimate before the measurement update correction.
- $\hat{P}\bar{k}$ is the prior error covariance.

The first step of a Kalman filter is to make a initial estimate and find the prior values ($\hat{x}\bar{k}$ and $\hat{P}\bar{k}$). We use these prior values in our measurement update equations. In this second set of equations, we really find $\hat{x}\bar{k}$. Also, we find $\hat{P}\bar{k}$ which is necessary for the future estimate, together with $\hat{x}\bar{k}$.

In our case, the Kalman filter will be used to know the device's orientation at any time. This means knowing three thing: roll, pitch and yaw. This is how the Kalman filter will help us:

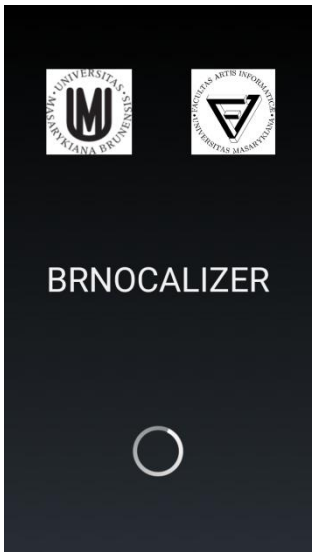


3. Schematics of an IMU built with a Kalman filter

2.2.5 The "Brnocalizer" application

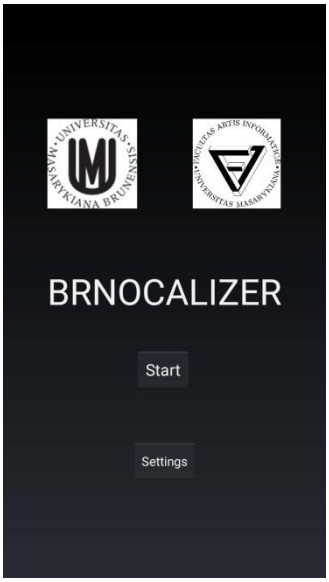
In the end, the application is composed of four different android activities:

- A connection activity in which the device is connecting to the associated server.



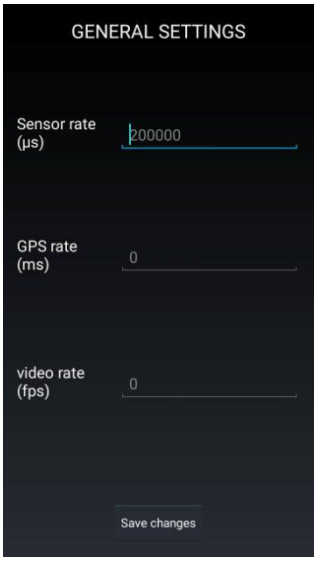
4. Screenshot of the connection activity

- A "home" activity in which the user can choose whether to set up the different parameters of the experiment or to start it with the default values.



5. Screenshot of the home activity

- A setup activity in which the user can change the acquisition rate of the sensors data, GPS data and change the frame per second rate of the video feed.



6. Screenshot of the setup activity

- A working activity that is the core of the application. In this activity, the different pieces of information are collected and then sent to the server.

7. Screenshot of the "working activity"

Those activities are backed-up by some classic java class that are storing parameters, establishing connection or sending data to a server. The communication between the different activities and those classes is done through a Service. This Android technology is offering a much more responsive application and a way to ensure the good execution of the application.

3- Appendices

3.1 Link to the project repository

The project was designed to be completely free of use and the source code, properly documented, can be found at:

https://github.com/wataburden/Brnocalizer

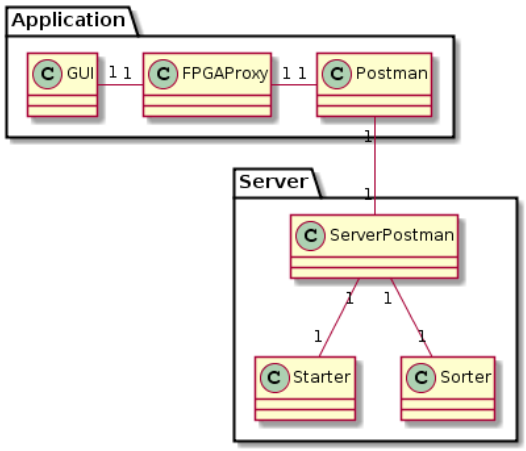
This link gives access to the source code of the application and the source code of the java server needed to transfer the data collected on the app to a computer and the project's javadoc.

3.2 Detailed class diagram

This schematic is describing how the project was built. There are two main objects in this project: the android application and the server. These objects are communicating with each other using their "Postmen".

The application is composed of a GUI allowing the user to navigate through it and an FPGAProxy simulating the presence of the server.

The server has a Sorter that is designed to sort the received message and execute some actions linked to those messages.



8. Project's detailed class diagram

3.3 Android application diagram

The Android application was designed using the MVC (Model View Controller) Pattern. Thus the application has 3 different packages:

- Activities: contains all the tools describing the GUI.
- Worker: contains the data classes needed to the good execution of the application.
- Communication: contains the classes needed to ensure a good communication with the server.

Package Activities

Welcome: Activity responsible for connecting the device to the associated server.

StarterScreen: Activity from which you can access the setup activity or start the experiment.

SetupScreen: Activity in which you can change some parameters of the experiment such as sensors' acquisition rate or GPS's acquisition rate.

SensorScreen: Activity responsible for data collecting and video recording. This class is associated to the class Protocol (package communication) in order to have a standardized way of sending data to the server and avoid any error.

SendingVideoScreen: Activity responsible for sending the video to the server and wait until that action is completed.

Package Worker

Relay: Service linking the activities to the other java class of the application. This android technology is used to ensure the data's safety during the execution of the application.

Settings: Java class containing the different parameters' data. This class is used to keep all of the parameters at one place in the application and prevent some data errors.

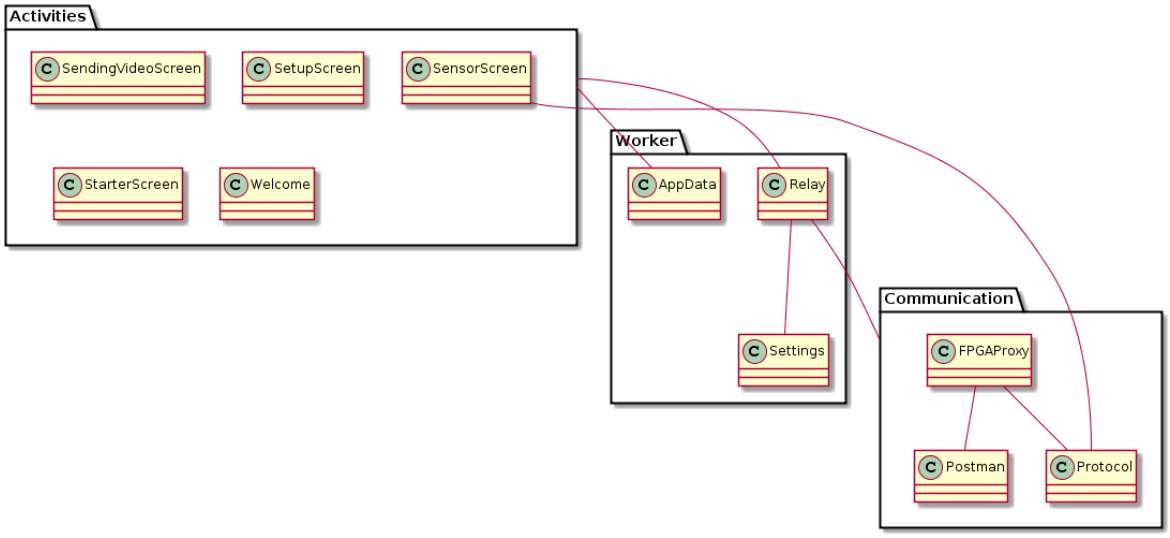
AppData: Java class gathering all the different variables needed during the execution of the application. Doing this kind of class prevents any errors in the inner-app communication.

Package Communication

FPGAProxy: This java class simulates the presence of the server for the application. The real work of this class is to ask the postman class to send some data to the server.

Protocol: This java class is establishing a protocol to send data to the server to prevent any errors in the encryption/decryption of the data.

Postman: This class is doing all the communication work by connecting the device to the server and then sending data to this server.



9. Application's class diagram

3.4 Sources

Accelerometer: *http://www.dimensionengineering.com/info/accelerometers*

Gyroscope and magnetometer: *Indoor Positioning using Sensor-fusion in Android Devices by Ubejd Shala and Angela Rodriguez (http://hkr.diva-portal.org/smash/get/diva2:475619/FULLTEXT02.pdf)*

Kalman filter: *http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies*