Accessible Indoor Localisation for Android

Shubham Kapoor (014601220),
Sebastian Hojas (014570704)

January 2016

1 Introduction

Locating ourselves has been a curious task for mankind for long. With the dawn of today's digital era various new wireless positioning techniques, such GPS, GLONASS, E-CID, OTDOA etc., were developed. These techniques provided overwhelming accuracy with a limitation: None of them worked accurately indoors and needed ”clear” sky view for attending there promised accuracy. Moreover complex and diverse indoor environments make tracking indoor a tedious task. However increasing popularity of commercialized location-based advertising, indoor logistics, analytics etc. motivated industry to work on unexplored area of indoor localization. Various new techniques were devised for same ranging from magnetic to Wi-Fi positioning.

We developed a handy mobile Android application called *Localiser*, which uses Wi-Fi positioning technique at its heart and calculates the devices's position in an indoor location with an accuracy of 5 meters. This application further gives the user details of point of interests near his/her location, such as nearby class rooms. Our app has implemented all native Android accessibility features, which would allow a visually impaired person to still use our app fluently and orient through campus.

2 Features

Some of the features of our application we want to highlight are:

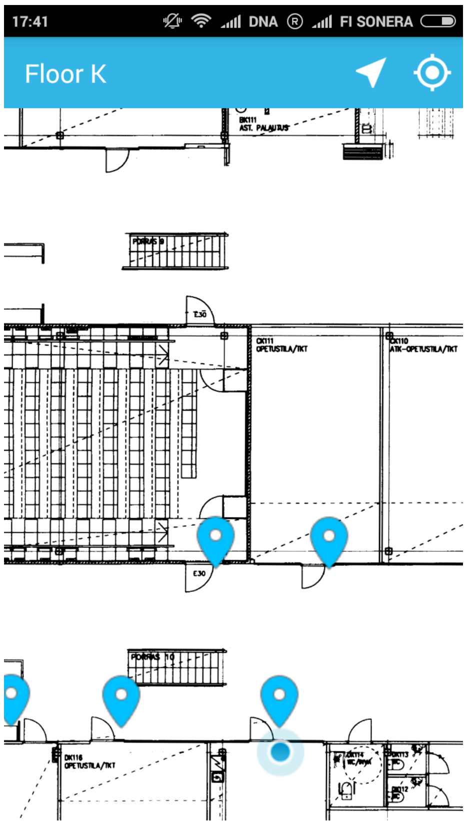


Figure 1: Screenshot of the application showing the current location and closest classroms nearby.

2.1 Show current Location

The application automatically shows the current location and updates it. Additionally, the user can use the button on the top right corner to center the current location on the map. The map will automatically change the displayed floor if the location changes. The user can scroll and zoom the app to get a better overview.

2.2 Nearby points of interest

This button is located next to current location button. Pressing this button displays nearby landmarks relative to current position of the user on the map. In our current version of our application we have defined around 100 classrooms and a couple of emergency exits (as seen in figure 1), but this can be easily extended with more information about the building.

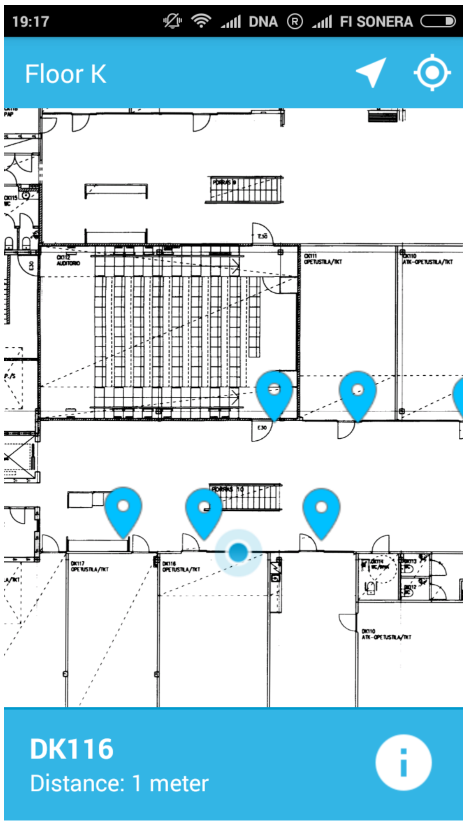


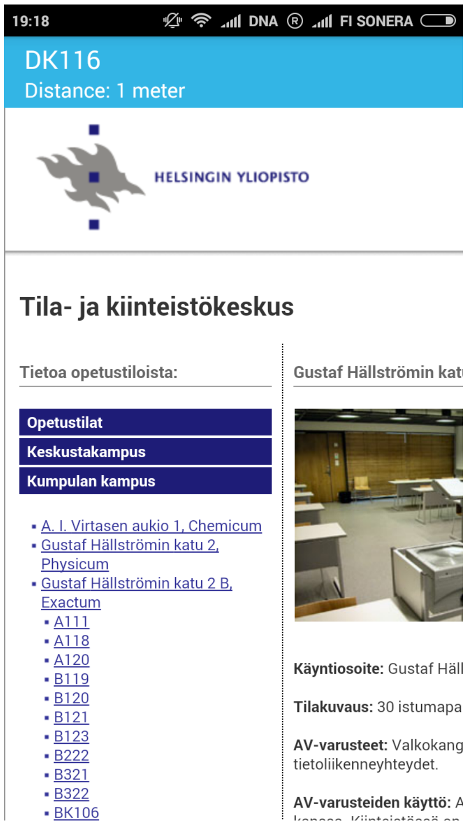
Figure 2: After a user tapped on a point of interest it shows its name and the distance to the user.

2.3 Vertical position of user

The title of the current application displays the current floor number.

2.4 Details of landmarks

If a user presses one of the displayed nearby landmarks, the user gets information about the landmark, as illustrated in figure 2. This includes its name and distance from the user. The user can then access the web-page of the defined landmark by clicking the information symbol which is at lower right corner of screen, as shown in figure 3. Not all landmarks currently have a web-page associated with them. Hence the information symbol does only appear for classrooms that have a site on the website of the University of Helsinki.


19:18 DNA FI SONERA

DK116
Distance: 1 meter

HELSINGIN YLIOPISTO

Tila- ja kiinteistökeskus

Tietoa opetustiloista:

Opetustilat
Keskustakampus
Kumpulan kampus

- A. I. Virtasen aukio 1, Chemicum
- Gustaf Hällströmin katu 2, Physicum
- Gustaf Hällströmin katu 2 B, Exactum
  - A111
  - A118
  - A120
  - B119
  - B120
  - B121
  - B123
  - B222
  - B321
  - B322
  - BK106

Gustaf Hällströmin kat

Käyntiosoite: Gustaf Häll

Tilakuvaus: 30 istumapa

AV-varusteet: Valkokang tietoliikenneyhteydet.

AV-varusteiden käyttö: A

Figure 3: Some rooms link to the website of the University of Helsinki to show further information. This website includes information about the room size, its accessibility characteristics and contact information.

2.5 Accessibility for visually impaired users

We have implemented native Android accessibility features that allow users who need assistance to use the application. We have successfully tested the application while being blindsided in accessibility mode Talk-back. We have made the following design decisions:

- Every button and action has a content description that will be used by the Android Talk-back feature.
- This also includes all elements on the map. The current location will always be in the middle of the phone screen. If the users slides with his/her finger over the screen, nearby locations will be marked by vibrational and audio feedback as defined by the accessibility best-practise by Google. This also includes the distance in meters to the point of interest.

For further development we would suggest to use sensor data from the device's compass to be able to give the user information about the relative direction of the point of locations around the device.

3 Algorithm

The algorithm is the heart of the application that predicts the user's location with a probability p for a given fingerprint. This algorithm is logically split into two parts:

- **Fingerprint comparison (comparator)**: The comparator takes two fingerprints, compares them and calculates the similarity. We have observed that is the same problem class as string similarity calculation. Thus, we mainly implemented variations of known string comparison algorithms.
- **Location calculation**: Knowing the similarity of all fingerprints in the database to the reference scan, it is now easy to calculate the location. This calculation can be a simple closest match or an elaborate probability calculation.

3.1 Fingerprint comparison

We have implemented two different comparators:

- **Intersection matching**: Considers only the intersection of two fingerprints and calculates the similarity of observed signal strengths of both data sets. This is a simple approach with a higher error rate because it favours fingerprints with small subsets and matching signal strength.
- **Cosine matching:** The cosine comparator considers the whole subset and therefore delivers more accurate results. It calculates the cosine similarity between the two fingerprints. We use this comparator by default.

3.2 Location calculation

- **Nearest Neighbor**: The algorithm matches the fingerprint with the highest similarity and takes its location.
- **k-Nearest Neighbors**: The algorithm matches the k nearest neighbors and calculates the coordinates based on the weighted location of all fingerprints.
- **Average**: For localisation, we wanted to implement a high/low pass filter that uses the median to improve accuracy. For simplicity, we implemented an average algorithm that takes n measurements, calculates n locations with the k-nearest-neighbors-algorithm and takes the average. This reduces average jumps (especially to avoid sudden changes of the Z value). The high fingerprinting frequency of our test device allowed us to still update the location approximately every three seconds, which is enough for our use case.

The next step would be to extend the average algorithm and create a more complex probability model (i.e. a Kalman filter) which also considers probability of changes in time and input from the motion sensors (gyroscope, compass and accelerometer). This way, we believe, we could increase the accuracy tremendously.

4 Database

The matching algorithm uses a data set of 1592 fingerprints. We created an individual Android application for the collection of fingerprints. It followed this simple workflow:

1. Define a path with n steps and confirm.
2. The application will then ask the user to walk through the path, stopping at every defined step.
3. While the user walks, the application takes as many fingerprints as possible and calculates an approximated location based on the assumption of linear walking speed between each stops, as shown in figure 4.
4. At the end of the path, the user can save the appended database file on the SD card. The new dataset is then ready to be manually downloaded from the phone.

The application can be started by launching *CollectFingerprintsActivity* and needs *android.permission.WRITE_EXTERNAL_STORAGE* which is temporarily disabled for the production. We chose to create an Android application to enable the collection of fingerprints with a mobile device.

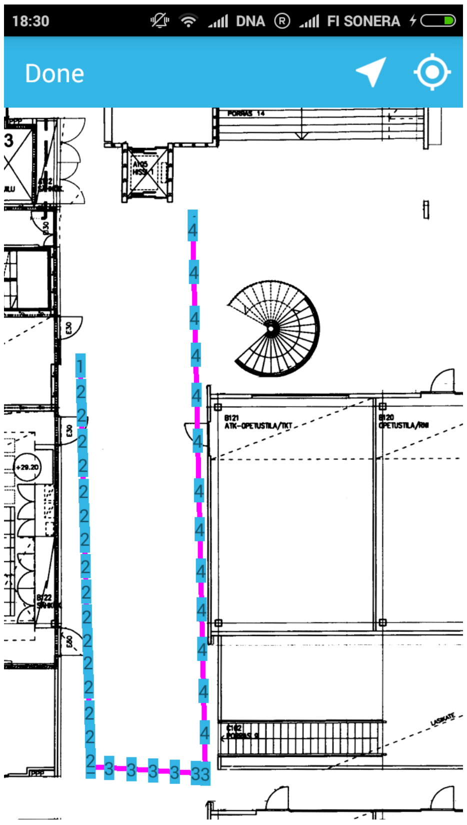


Figure 4: Example how we used our Collector application to massively collect fingerprints

5 Software Architecture

The Android GUI activities use the *LocaliserController* to register for location events. When initiating the controller, an algorithm can be chosen. Every algorithm can use a different comparator, that defines how fingerprints are being compared to each other. This allows the activity to easily choose the used algorithm and its comparator.

Usage of a controller with the *k-nearest-neighbors* algorithm with the *Cosine* comparator would be the following:

```
AbstractLocaliserAlgorithm algo = new
 ↪  kNearestNeighborsAlgorithm(new CosineComparator());

try{
    LocaliserController testController = new
     ↪  LocaliserController(algo, context);
    testController.registerForLocationUpdates(this);
}
catch(LocaliserController.NoWIFIException err)
{
    // TODO error handling: WIFI was disabled
}
catch(IOException err)
{
    // TODO error handling: Could not read database
}
```

This dynamic architecture allowed us to change location algorithms and comparators easily and even test different combination simultaneously.

6 Testing

We have developed our application for the following platforms. We would be available for the demonstration of the application on our tested device.

6.1 Platform

We have successfully tested our application on a *Xiaomi Redmi 1S device.* The device has a 1.6GHz Quad-core Qualcomm Snapdragon 400 processor with Cortex-A7 core. The device has 1 GB of RAM and 8 GB of FLASH internal memory. This device supports standard Wi-Fi 802.11 b/g/n. Our device runs on android Kitkat with API level 19. While we developed our application for API level 14 and above, we have not been able to test our application on different Android device and different API levels.

6.2 Accuracy

The current version of our application gave us an accuracy of about 5 meters with the above mentioned test device. We have covered the area in Exactum as defined for the first assignment, without the locked computer lab. We also tested in-room localisation in the 24/7 computer-lab and have been very satisfied with the accuracy in closed and open spaces.