IN4254 Smart Phone Sensing - Final Report

Aniket Dhar (4615808) and Sergio Soto (4627288)
Phones Used: Samsung Galaxy A5 (Android 6.0.1),
Samsung Galaxy Nexus (Android 4.2.2)

Abstract— **Indoor localisation is gaining grounds in the past decade, both in research and industry. With the introduction of WiFi and variety of sensors in the mobile phone, it has become possible to use an average smart phone to track movement and location in indoor environment with high accuracy. In this report, we present and analyse an Android application which makes use of WiFi strength along with accelerometer and magnetometer sensor data to accurately determine the location of a mobile user in a predefined indoor map.**

I. INTRODUCTION

The final assignment for the course **IN4254 Smart Phone Sensing** entails development of an Android application (app) which should be able to determine the location of a mobile user in a predefined indoor environment, in this case the 9th floor of the EWI faculty in TU Delft, divided in 20 cells. The application uses WiFi RSSI signal data from nearby access points, accelerometer values for motion detection and step-count, and magnetometer values for determining the direction of motion.

The app first estimates the initial belief of user location through Bayesian filtering of the WiFi RSSI (Received Signal Strength Indicator) of the available access points in different locations, which have been trained, classified and then stored in the app. Once an initial belief of the location is obtained with sufficient accuracy through multiple iterations, a particle filtering approach is utilised along with a motion model to track the movement of the user as he or she walks within the indoor environment, effectively narrowing down the current position within the space. The app in its complete form is able to track the user's movement and compute with high accuracy, his current location.

The report starts with a brief description of initial belief estimate through Bayesian localisation; then it delves into the intricacies of particle filter implemented. The report also briefly describes the individual work-load for completing this project and the future paths that can be taken to further improve it.

II. BAYESIAN LOCALISATION:

Bayesian localisation aims at determining the location of a mobile object using the Bayes law of probability. After acquiring characteristic data (RSSI from access points in our case), this method calculates the probabilities of being at each of the 20 cells. It creates an initial belief, and then iteratively acquires more samples and adjusts the position until a certain probability threshold is reached.

A. Data Collection

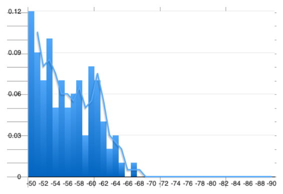
First, the app gathers training data to train the classifiers. Then these classifiers are used to optimally define the labels for user data on the go. All our data acquisition and processing has been done on-line in the phone, in the app itself.

During the data acquisition and training process, the app scans all Access Points visible to it at each radio cell location. These values are stored according to their MAC address in decreasing signal strengths. 120 samples are taken per cell, and stored in data structures in the phone for further processing. Training data was acquired on different days and facing different directions, covering as much room as possible. Thus, the training phase takes multipath fading and presence of varying obstruction objects into consideration. Since the samples are taken one after another, there is no fixed sampling rate since this varies on the number of access points available at each cell. However, it can be estimated to about 1 to 2 samples taken per second.

B. Data Processing

The RSSI signal strength for each MAC address is converted to discrete RSS levels (0 through 255) and initially saved in a Map data structure as key-value pairs. Then the histogram of the signal data is calculated and represented as a Probability Mass Function (PMF). A Gaussian fit is formulated over the PMF data set which consists of mean and variance pair for each Access Point corresponding to every radio cell in our training environment. Computing the Gaussian distribution and storing it in this way reduces space in memory and adds granularity to our data for the RSS values that weren't captured during data acquisition. The Gaussian fit for every AP is computed when the user presses a button after data from all rooms has been collected, and these are stored in the phone memory through serialization.

C. Radio map


(a) Cell 14

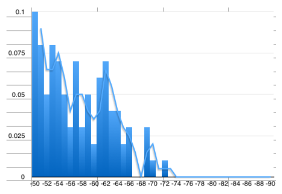

(b) Cell 15

Fig. 1: Similar radio maps of closely placed cells

The result of processing the data is having a radio map for each cell, where the RSS of each access point is represented as a histogram. In our case, these were estimated as normal distributions. For the most part, the data processed resulted in distinctive radio maps. However, for some cases like C14 and C15 (See figures 1a and 1b), due to the layout of the rooms, the radio maps look very similar and further techniques are needed to distinguish between them.

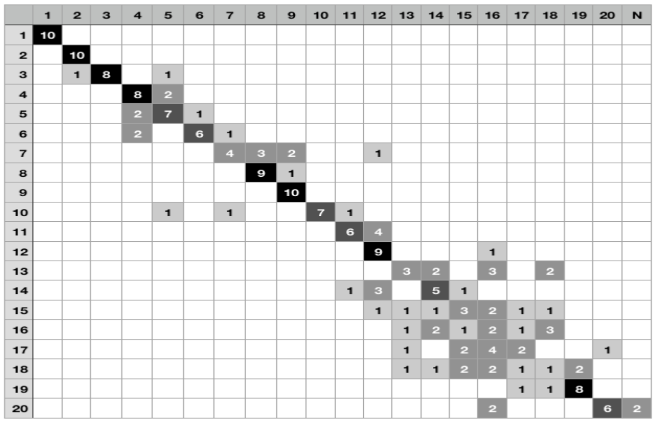
As it will be discussed in section IV, our implementation does not filter temporal access points or duplicates.

D. Testing

During testing, the app scans the WiFi signal strengths and the Access Points obtained are classified into labels by implementing Bayesian filter on the training data. The Access Points are sorted in decreasing order of their RSS strength. The strongest 3 Access Points are considered for the testing process. A Perception Model is created and the model is updated iteratively to find the final position. Initially the Perception Model imparts equal probability to all the cells where the user might be located. Then, this prior belief is updated in each iteration depending on the probability of the user being located in a certain cell, given the strength of the strongest Access Point scanned. The process is repeated for each of the 3 access points to obtain the location of the user, giving a final probability for each cell. The whole process is repeated multiple times to refine the resulting probabilities.

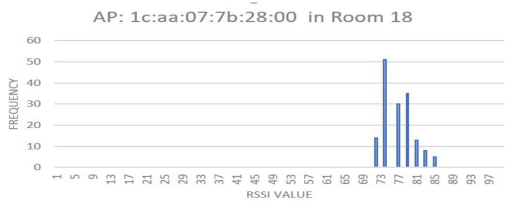
E. Results

We performed 10 tests in each cell. Since we found that for some rooms further iterations result in a better estimate, and for some it results in degrading it, the results presented are taken from a single iteration of the Bayesian filter. They can be seen in the confusion matrix in figure 2. It can be seen that for the most part, the initial belief follows the main diagonal. This means that for the most part, the results are correct on the first iteration. However from cells 13-20 (where there is the biggest concentration of rooms next to each other), the quality of the initial belief is drastically degraded. A special case is cell 20, where due to the physical characteristics of the room, some scans will detect no access points at all, resulting in each room having the same 5% probability.

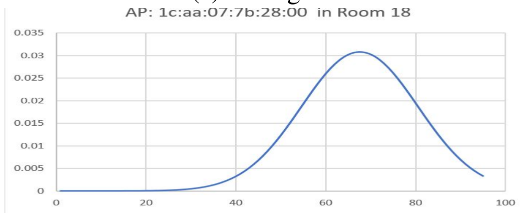


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	N
1	10																				
2		10																			
3		1	8		1																
4				8	2																
5				2	7	1															
6				2		6	1														
7							4	3	2			1									
8								9	1												
9									10												
10					1		1			7	1										
11											6	4									
12												9				1					
13													3	2		3		2			
14											1	3		5	1						
15												1	1	1	3	2	1	1			
16													1	2	1	2	1	3			
17													1		2	4	2			1	
18													1	1	2	2	1	1	2		
19																	1	1	8		
20																2				6	2

Fig. 2: Confusion matrix



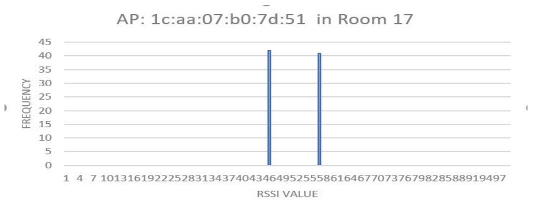
(a) Histogram



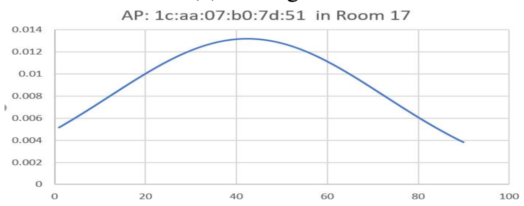
(b) Gaussian Fit

Fig. 3: Histogram and Gaussian fit for room 18

The difference in accuracy for different rooms can be accurately expressed through the following two Gaussian fit model examples for two rooms. While in Room 18 (figures 3b and 3a), the particular Access Point has almost a normal distribution of RSSI strengths, the one in Room 17 (figures 4b and 4a) shows very discrete distribution, and hence the Gaussian model developed upon it doesn't characterise properly its actual distribution graph.



(a) Histogram



(b) Gaussian Fit

Fig. 4: Histogram and Gaussian fit for room 17

Different environmental conditions present during both testing and training like number of Access Points broadcasting, people and devices around the user, and changes in location or magnitude of various obstructions (like doors open or closed) also influence the accuracy of results.

In conclusion, the Bayesian localisation model provides a decent initial location estimate upon which our Particle Filter Localisation can start from.

III. PARTICLE FILTER LOCALISATION

The particle filter localisation process needs a layout map of the testing environment and a motion model that estimates how the user moves, including considerable noise estimations in measurement. The following sections delve deeper into the implementation of the localisation model.

A. Location map

A layout of the 9th floor of the EWI building in TU Delft was created according to scale (Figure 5. The map was divided into a 13 x 3 grid of objects we call Blocks, which hold the range of coordinates they represent in the map, as well as the property of whether this corresponds to a cell or not, and the cell

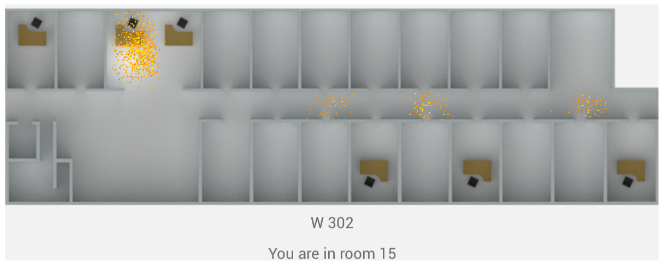


Fig. 5: Map layout and particles

number when applicable. A scaling factor is used in order to translate distance from meters to pixels within the ImageView object holding the map as follows:

$$SF = \frac{MapWidth_{pixels}}{4_{meters} * 13}$$

B. Motion Model

The motion model deals with the movement of the user in each sampling interval. It estimates the distance covered by the user with direction after every step taken, with some noise considerations.

The main parts of the Motion Model are the Compass and the Step Counter.

1) Compass: The Compass in our implementation predicts the direction of motion by `Rotation Matrix` through the use of accelerometer and magnetic sensors. The raw data obtained from the sensors are smoothed down before being sent to the `Rotation Matrix` function, provided by the sensor manager class, to remove rapid changes in the output direction.

The orientation angles are clustered into 8 cardinal directions. This implies that the user can only move in angles multiple of 45 degrees. For example, the app will only consider the user moving either to the north, or to the north-west, and doesn't take into account angles in between. We decided to choose this method since the accuracy of the sensor of our phones was not producing an output consistent enough for us to deal with small variations in direction.

In our app, the north is adjusted exactly as according to the map. That is, the top of our map is north. In order to adjust for this, the user must calibrate the compass such that when facing opposite to the elevators, the sensor output must read as close to ”N 23” as possible.

2) Step counter: The Step Counter counts the number of steps taken by the user. Since the Sensor `"TYPE_STEP_COUNTER"` is not supported by our Android devices, it was imperative to implement our own, based on just accelerometer data. Our Step Counter mainly consists of a Step Detector which detects if and when the user takes a step. The Step Detector uses a moving window average of accelerometer values to initially calculate a normalised global acceleration vector. *Then the component of current acceleration in the direction of the global vector is calculated and a velocity estimate is evaluated from it. Velocity estimate is compared to certain speed threshold as well as timing threshold constraints are checked to determine if a step has been taken.*

C. Particle movement

When a step has been detected, the position offsets in both x and y directions are calculated according to the direction of motion obtained from the Compass. This takes into account the stride length of the user, which is also made adjustable through our app interface. Once the step offsets are calculated, our particle distribution is updated within the map according to the scaling factor described in III-A. The exact new position of the particle (which includes a noise element accounting for sensor inaccuracies) and the behaviour it will have are described in the following section.

D. Particles

We define a (configurable) maximum number of particles of 1000, which in our case was sufficient to accurately locate and display the location. Our Particle object holds the following characteristics: An (x, y) coordinate representation in the map, *and a parameter we call age, which ranges from 0 to 1. In the map, the colour of particles describe their age. "Young" particles start with a yellow colour, turning gradually into red as they age.*

E. Particle filter

The particle filter work flow is the following:

1) Once either a Bayesian new belief or an iteration after it has calculated a probability for all the cells, the maximum number of particles is divided among all Blocks that represent a cell, proportionally to their probability, with their individual (x,y) coordinates calculated randomly within the boundaries of the cell. *An initial age of 0.5 is given to them.*
2) Whenever the user moves and a step is detected, a new coordinate is calculated based on the offset distance calculated by the motion model, plus a random noisy distance in both axes.
3) If the (x,y) new coordinate of a particle falls either on a Block which doesn't correspond to a cell, or out of the map boundaries, we say the particles *dies*, and it's accumulated to a bucket of dead particles. Otherwise, the particle's *age* increments by 0.01.
4) *Once all cells have been moved, a special group of particles is calculated, corresponding to the eldest 200 ones.*
5) Finally, *the dead particles are revived. Their new age is set to 0.0, and their new (x, y) coordinates are set to a random (confined) distance from a randomly chosen particle from the eldest group.*

F. Location narrowing

Once the user has taken a few steps, the particles rapidly start converging. *The final estimate of the user location is narrowed down by averaging the location of the eldest 200 particles. Then the Block corresponding to these coordinates is found and, if it is a cell, the cell number is displayed to the user and the room in the map is "lit up" for a more visual indication, as seen in Figure 5.*

G. Results

We performed 10 tests on the Step Counter to check the accuracy of steps detected. On each test we took 100 steps, keeping the phone in our hands at chest height, without much vertical movement. The step counter displayed 7 steps extra over all 10 experiments giving an error rate of (7/(10*100)) = 0.7%.

For our particle filter tracking, once the particles converged, we went through random paths to check how the particles followed us. Starting from the cell of particle convergence, we moved to each of the remaining 19 cells randomly. The error is calculated as :

$$\frac{\sum abs(index\_actualcell - index\_detectedcell)}{Number\_cells * Number\_tests}$$

15 such tests have been done and the error is found to be 0.223%

IV. CHALLENGES AND INNOVATION

One of the biggest challenges with the Bayesian filter is mitigating with temporal access points. A temporal access point could be caused due to a "hotspot" WLAN, different routers even on other floors intermittently showing up on our measurements, and duplicated networks with different access points. Our app gathers and generates a PMF table for each access point. A missing AP during either training or testing phase will result in a table being ignored, or a table not being created, respectively. During testing, we perform a lookup for the top 3 APs read. However, if a table is not found for the specific AP, this RSSI value will be ignored but still "consume" one iteration. This means it won't affect the result, but it will impact the precision of the accumulated iterations.

Another challenge we had to overcome was having to implement a step counter ourselves. Since achieving a perfect count of steps would have been a very time consuming task, we instead address the issue of inaccuracy on the particle filter side. By moving particles at random distances and with a moderate amount of particles, we ensure that at least some of them move the correct distance, which in turn guarantees they will age as they will be more likely to survive.

Speaking of particles, one last challenge we faced was the inaccuracy of the initial Bayesian belief. To mitigate this, we do two things: One, we make sure that every cell has a minimum probability of being the correct one (2%), regardless of the Bayesian output. Two, instead of positioning the particles at a coordinate given in part by the Bayesian estimate, we create them next to the oldest ones, regardless they are correct or not. The reasoning behind this follows in a certain way the behaviour of some animal species like wolves, where the pack always follows the eldest. Granted, calculating by other means the position of a new particle could result in a better estimate (e.g. closer to the user's location) and thus require the user to walk less before particles converge. However by following the oldest particles, regardless how likely they were at the beginning to be the correct one ensures that eventually when they converge, the result will be correct, and the Bayesian estimate provides a good starting point to minimize convergence time. In the end, it's a trade-off between time (or distance walked) and accuracy. Since the purpose of the app was to locate the user with precision, we opted for the latter.

V. INDIVIDUAL CONTRIBUTIONS

A. *Aniket*

Implemented the basic structure of the app layout and the main activity; implemented permission issues and sensor data handling for KNN, Bayesian and Particle filters; worked on logging and data serialisation in phone memory; implemented direction sense and Step Counter; jointly worked on Bayesian and Particle filter algorithms.

B. *Sergio*

Implemented original assignment's KNN algorithms for movement detection and location tracking. Implemented the algorithms, classes and data structures for Bayesian and Particle filter. Designed the floor map and boundary checking. Implemented particle generation, movement, behaviour and display on map.

VI. POSSIBLE FUTURE WORK

Had we had more time, there are some things we would have liked to implement in our app:

On the Bayesian filter side, we would have liked to come up with an alternative to the Gaussian fitting that more accurately describes the distribution of the RSSI levels. One idea was to approximate a Bezier curve to the PMF distribution, in order for it to be continuous while minimizing the amount of data stored. For cells 13-20, we would have liked testing the result of taking a moderately higher amount of samples in order to improve accuracy in that area.

On the Particle filter side, we would have liked to make the mechanism more modular. The current implementation is time consuming in the sense that a new map would have to be generated if the system was to be implemented elsewhere. If the architecture doesn't follow a square grid like the EWI building, Blocks would need to be approximated as triangles in order to fit different layouts. Also, if the Bayesian was more accurate we could have made new particles appear at the correct estimated location, instead of following the eldest ones.

Regarding new ideas, we originally created a (minimal) 3D model of the 9th floor, with the idea of being able to "move" within the 3D map. Expanding further, the idea of using the phone's camera in an augmented reality kind of way is not too far-fetched, and with image processing and computer vision algorithms the accuracy and location speed could further be increased.

VII. CONCLUSION

Bayesian filters are a very robust mechanism to analyse and categorize noisy input data and compare it to a trained model. On the other side, particle filters are a very effective mechanism to narrow down and further categorize a dynamic and changing system over time. By combining both, and by using previously learned concepts like KNN, we successfully created an app for Android that tracks a user's position. Understanding the complexity of how localization works added a whole new set of possibilities to our engineering knowledge, and made us appreciate the systems we use every day and take for granted. We certainly won't be so impatient next time Google Maps struggles calculating the direction we're facing to from now on.

REFERENCES

[1] Android Developers Documentation (https://developer.android.com/training/index.html)

[2] Bayesian filtering for indoor localization and tracking in wireless sensor networks (https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/1687-1499-2012-21)

[3] Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond (http://www.dsi.unifi.it/users/chisci/idfric/Nonlinear_filtering_Chen.pdf)

[4] Paramvir Bahl, Venkata N. Padmanabhan, ”RADAR: An In-Building RF-based User Location and Tracking System”, IEEE INFOCOM 2000

[5] Anshul Rai, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, Rijurekha Sen, ”Zee: Zero-Effort Crowdsourcing for Indoor Localization”, Mobicom ’12, Istanbul, Turkey, August 2012

[6] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, ”A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”, IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 2, FEBRUARY 2002