IN4254 Smart Phone Sensing Report

Group 16: Unsure

Alex Berndt
4698959
a.e.berndt@student.tudelft.nl

Ernst Mulders
1504673
e.a.mulders@student.tudelft.nl

1 BASIC INFORMATION

Project chosen: Project 1 - Localization and Particle filter
Phones used:

(1) Motorola G4 Plus Android 7.0 for Bayesian Localization
(2) OnePlus One Android 6.0.1 for Particle Filter

Points of **innovation** are marked with a **star** (^{*i}).

The code used for our app can be found at:
https://github.com/ernstmul/SmarphoneSensing

2 BAYESIAN LOCALIZATION

2.1 Data Collection

Data collection is performed by sampling the RSSI levels of visible Wifi access points at each cell. A histogram (see Figure 1) is created for each cell, for each visible access point, with the bins representing the RSSI value seen for the specified access point. Data collection therefore only yields the frequency data of each access point seen at each cell. Section 2.3 details how the Gaussian PMFs are obtained from this data.

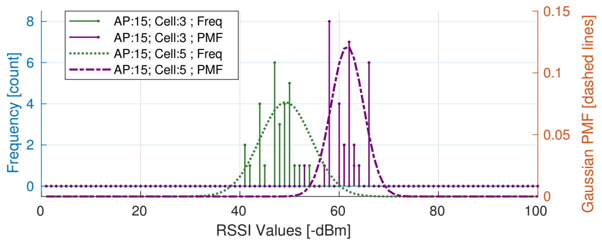

Figure 1: RSSI data and Gaussian fitting - Cell 3 and 5

2.2 Training

Training Description

(1) Training is done in the 4 cardinal compass directions (NSWE). The idea is to eliminate the effects the human body has on the received RSSI signal.
(2) Data was obtained on different days at different times. This is done to eliminate temporary or weak access points which are not seen frequently. For this, we have made it possible to merge data over multiple days.
(3) Each of the 19 cells must be trained during one training session. This is critical because we only save frequency data of the received signals, therefore each cell must be trained the same amount of times.
(4) Training was done on 6 separate days, between 10:00 and 18:00 resulting in 64 measurements per cell.

Merging Training Data

Merging data is done by simply adding the frequency data of each access point for each cell of the two data points. If an access point is only present in one of the training sessions it is not added. This ensures temporary and weak access points are discarded.

2.3 External Data Processing

The training data is then converted from a frequency based histogram to a Gaussian PDF using a curve-fitting tool in MATLAB. This PDF is then integrated across its bins to form a PMF, which is logged to the device for on-board calculations. Note that a PMF must be used in the bayesian filter since a PMF's *discrete* probabilities always sum to 1 (this is not necessarily the case for PDFs).

2.4 Parallel Bayesian Weighting ^{*1}

Once the PMFs are saved to the Android device, it can use the PMF entries based on a current Wifi measurement to locate itself. This is done by considering the current RSSI level of each access point and performing Bayesian filtering to eventually converge towards a most probable current location.

It was found that a parallel Bayesian filter architecture (see Figure 3) is more reliable than a series equivalent. This is because a parallel Bayesian filter isn't affected by initial, inaccurate RSSI values which could throw the belief statement off and never allow it to recover.

When summed together, the obtained probability for each access point is *weighted* according to the expected reliability of the signal. The likelihood of each cell c in **P** is then determined as:

$$\mathbf{P}(i, r, c) = \sum_{i=1}^{c} w_i(r) P(i, c) \tag{1}$$

where P is the Bayes' probability of cell c at access point i given measurement level r. The weight $w_i(r)$ of equation 2 is proportional to the approximate variance of the signal at different RSSI levels (and therefore distances) from an access point. The parameters P, L_0 and η were determined by measuring the signal variance of measurements at different RSSI strengths of a single access point and fitting Equation 2 using least-squares regression. This is shown in Figure 2.

$$w_i(r) = PL_0 + 10\eta \log_{10}(r) = 2.03 - 0.031 \log_{10}(r) \tag{2}$$

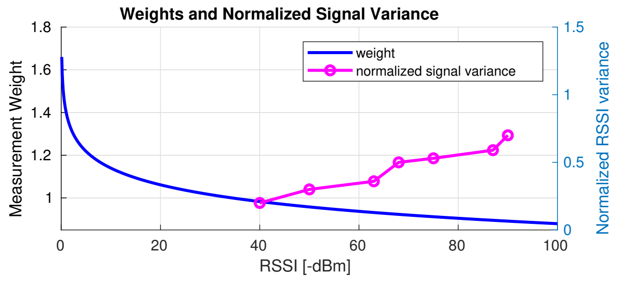

Figure 2: Measurement Weight and Signal with Shadow and Multipath Fading

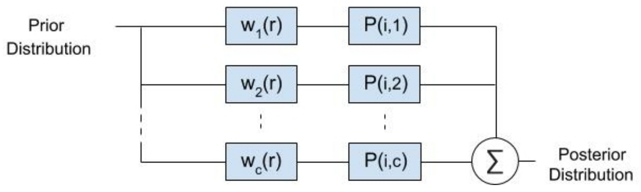

Figure 3: Schematic of Parallel Bayesian Filtering

2.5 Implementation Challenges

Despite the bayesian filter being relatively simple to implement, we faced the challenge of ensuring the most recent Wifi scan data was used in our filter. We suspect that the Android operating system keeps a log of Wifi measurements and doesn't update it with each scan to save energy. In order to overcome this problem, we perform a scan sequence which ensures 10 unique Wifi scans. We then apply our filter on the last 5 unique scans and take the most frequently seen cell value as the current location. We found that this ensured that we performed localization using the actual current Wifi scan values.

2.6 Testing Methodology

Testing was done by standing in each cell with a random orientation and performing a localization scan. This was repeated 15 times for each cell. The confusion matrix in Table 1 shows the results obtained for each cell.

2.7 Results

As can be seen from the testing results in Table 1, the implemented bayesian filter is very accurate. The average rate of correctly identifying a cell is 98.15%.

2.8 Discussion

The results can be deemed to be satisfactory. A cell was only incorrectly identified 1.85% of the time. The key to obtaining this low error rate is the parallel weighted bayesian implementation, which accounts for bad signal affects by weighting stronger, more reliable RSSI signals more than weaker ones (thus accounting for the multi-path and shadow fading present in Wifi signals).

Another very important factor was performing 10 scans and only considering the result from the last 5. This ensured that the Wifi measurements used in the Bayesian filter were indeed the ones associated with the device's current location.

If the app could be given more direct control over the Wifi module, fewer scans would need to be performed, resulting in faster localization times. Using the aforementioned localization method, a single cell's localization takes on average 28 seconds.

Table 1: Confusion Matrix for Bayesian Filter

Cell	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
1	**0.93**	0.07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.07	**0.93**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	**0.93**	0	0.07	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	**0.93**	0.07	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0.07	**0.93**	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

3 PARTICLE FILTER LOCALIZATION

3.1 Motion Model

Our motion model relies on the *TYPE_STEP_DETECTOR* as well as the *TYPE_ORIENTATION* built-in Android virtual sensors. The step detector gives an approximation of the amount of steps a user has made with the device. The orientation is used to determine in which direction the user is walking. Since the Motorola G4 Plus does not have a compass, we used the OnePlus for the particle filter.

After starting the application, the step-counter needs to ‘warm up‘, it takes around 6 steps for the sensor to start triggering its onSensorChanged events. We obtain the quickest converging result when the particle filter is started after this warm-up period, however the particle filter eventually always finds the correct position.

Every time a step event takes place, we know a step has most probably been made. Through our user interface we've obtained the default step size in millimeters from the tester. This way we can estimate the distance covered by the user for each step.

In order to ensure the step is projected in the correct direction we gather the orientation information from the device's compass. The values the sensor provides are highly inaccurate, even after calibration of the compass. Despite this, since we are only walking in the 4 cardinal compass directions, the device can get a sufficient indication of which direction the user is walking in.

3.2 Map Implementation

The map implementation is based on Example 6 on the course Github[3]. Our map consists of three types of elements: walls, closed areas and accessible areas. Drawing these elements on the map canvas is processed by a function which takes the dimensions of the walls or closed areas in centimeters, as well as their position relative to the top left corner as input.

The centimeter values are converted into pixels by factoring them to the ratio in which the dimensions of the building fit the screen dimensions. This ratio is obtained by dividing the length of the long and short side of the building in centimeters by the screen height and width in pixels respectively.

3.3 Particle filter

Overview of Particle Filter Implementation

(1) On initialization of the particle filter 1000 particles are equally, randomly distributed over the accessible areas.
(2) When the device registers a step, a new position for all particles is computed based on a noisy motion model.
(3) Every particle is moved with the step size in the direction that the compass indicates. Artificial noise representing measurement and motion model noise is accounted for by adding zero mean Gaussian noise to the distance and orientation of each particle.

Particle Trajectory Collisions *[2]

After moving a particle we create a rectangle spanning the particle's old and new cartesian position. For this rectangle we check if it intersects any of the walls. By using this approach we significantly reduce the computational effort of having to check each intermediate position for a trajectory that collides with a wall. When a collision is detected, the particle is marked as 'dead' and added to a a list of dead particles.

After all particles have been moved, the dead particles are randomly re-distributed over the particles that are still alive.
In an earlier stage we have also implemented a collision method that, after detecting a collision, changed the variance of the distance and direction and tried to move the particle again until it succeeded. Although this worked, it was too computationally demanding for the OnePlus.

Refactoring Particles *[3]

It is possible for the particle filter to completely loose track of the user's current position due to erroneous measurements. In this case, the particles would never re-converge towards the user's actual location since the particles are all on the wrong side of the map. To account for this, we implemented a refactoring of 5 of the 1000 particles each step. This means that 5 randomly chosen particles are put somewhere on the map at each step.

This method allows the particles to re-converge in case they loose track of the user and is often used in actual particle filter implementations.

Motion Model Tuning

Since there is high variablity in user step sizes and compass direction accuracy, the particle's updated position is given an artificial gaussian noise both in its distance travelled and orientation. The mean of the added Gaussian noise is 0. However the variance was tuned as follows:

- Distance: variance of approximately 2.5 meters ensured that we could account for variable step sizes, while still ensuring trailing particles would disappear (due to collisions) when turning direction. This was tested in the long, narrow middle corridor seen in Figure 5
- Orientation was tuned to 5^o variance to allow particles to survive even if the user walks slightly skewed e.g. cell 12 to 15 (see Figure 5).

3.4 Stair Detection using SVM Classifier *[4]

Neither the Motorola nor the OnePlus has a barometer for floor detection. To overcome this, we implemented a linear Support Vector Machine (SVM) which classifies the IMU data of each step to determine if we are walking on stairs or walking on a flat surface.

Filtering Accelerometer and Gyroscope Measurements

A 20Hz low-pass filter is applied to the measurement data since all movement at higher frequencies is not associated with a person's gait [1].

Feature Selection

Six features were used for each step as suggested by [1] [2]:

X1	Max angular velocity in Y
X2	Variance of acceleration in Y
X3	Mean Energy value in X,Z
X4	Energy variance in X,Z
X5	Variance of acceleration in Z
X6	Mean of acceleration in Z

PCA Dimensionality Reduction

Principal component Analysis (PCA) is used to reduce the feature dimensions from six to two. PCA ensure that maximum data variance is kept for each dimension reduction. Figure 4 shows the two largest PCA components and the measurements in the 2D PCA subspace.

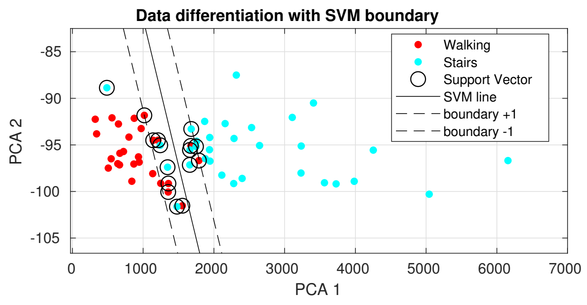

Figure 4: Walking and climbing/descending stairs in 2 PCA components

Training SVM

Training is done by collecting IMU measurements for walking normally as well as up/down stairs. Training the SVM yields a decision equation shown in Equation??. For a given feature set (obtained from a single step), $d(\vec{x}) < 0$ and $d(\vec{x}) \geq 0$ classify the walking as normal or up/down stairs respectively.

$$d(\vec{x}) = \vec{w}^T \vec{x} + \beta \tag{3}$$

Floor Detection

Due to sensor noise and walking variability, the line between walking normally and up/down stairs can vary a lot. To account for this, a floor change is registered only if in the last 12 steps, 4 of the steps were detected to be walking up/down stairs.

3.5 RANSAC Localization *5

Due to the nonlinear distribution of particles, finding the most probable current location cannot be done by simply taking the position based centroid of all the particles. This is even more evident in an example as shown in Figure 5. Note the three distinct clusters of particles at Cell 10, 16 and in the hallway.

To overcome this problem a RANSAC algorithm was implemented which uses a circular area as the fitting model. Multiple random particles are chosen and the one with the maximum number of particles within it's circular area is chosen. The centroid of the other particles within this particle's position's circular area is then used as the most probable current location (shown as the red dot in Figure 5).

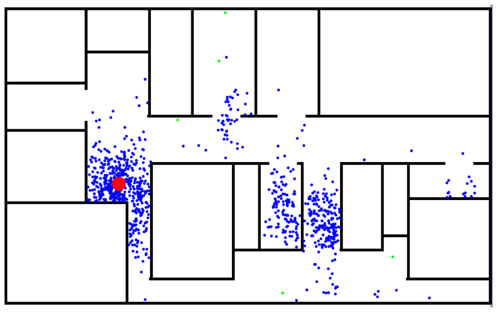

Figure 5: Localization using RANSAC with variable particle spread

3.6 Testing Results and Discussion

The Particle filter was tested by picking a random starting point on the fourth floor. As we walk, the particles are slowly start converging towards different areas on the map. The speed in which they converge is dependent on the amount of turns we take, or unique stretches we walk.

When entering the staircase some particles will end up in the hallway directly next to it (the entrance of the toilets) due to the variance. After walking down the stairs the floor change is detected, and we are successfully located on the third floor at the staircase.

Walking 15 unique routes with different starting and end points (each route had at least 2 turns) yielded 15 correct localizations when using the RANSAC localization method.

4 CHALLENGES AND SOLUTIONS

(1) A significant challenge of the project was the lack of computational power of our devices. Methods such as redrawing the canvas at each step on a different CPU thread ensured we could redraw 1000 particles fast enough to allow for normal paced walking without the app lagging behind.
(2) Another challenge was a seeming backlog in Wifi scan data. We suspect that the phone first provides us with some old scan results before actually doing new scans, despite the app specifically calling for a new scan to be done. This problem was solved by performing a set of 10 measurements for each localization request. We discard the first 5, and pick the most frequently determined cell from the last 5. This takes a little longer to compute, but ensures localization is performed with current Wifi data.
(3) Both of our devices lacked a barometer meaning that we would not be able to detect alttiude (i.e. which floor we are on). To overvome this, we implemented stairs detection feature which would analyze IMU (accelerometer and gyroscope) measurements at 200 Hz and classify the current step as being a normal walk or walking up/down stairs.

5 INNOVATION

The following innovative solutions were implemented to help overcome the challenges mentioned in Section 4. Innovations are marked with a $*^i$ through the report and correspond to:

$*^1$	Parallel Bayesian Weighting
$*^2$	Trajectory Detection for Particles
$*^3$	Refactoring Particles
$*^4$	Stair Detection using SVM Classifier
$*^5$	RANSAC Localization

6 INDIVIDUAL WORKLOAD

Almost all the work was done collaboratively. However, the members focused on the following:

- Alex implemented the RANSAC algorithm and SVM classification and derived most of the algorithms' main concepts
- Ernst wrote most of the Java code and ensured the algorithms were implemented efficiently

7 POSSIBLE FUTURE DIRECTIONS

Bayesian Localization

(1) If the app had direct access to the device's Wifi module, we could have more certainty that the current Wifi scan is indeed the one of the current location. This would significantly improve localization time.
(2) With more time, more training could be done, allowing our PMFs to accurately represent each access point's average measured signal strength at each cell.
(3) The code could be improved to make training less involved. This could be achieved by giving the user instructions on screen and automatically collecting the data.

Particle Filter

(1) With more powerful hardware, more particles could be generated and redrawing of the particles at each step could done faster. This could improve accuracy and allow users to walk faster.
(2) By applying a Gaussian Mixture Model (GMM) or a more sophisticated classifier to better discern between walking or walking up/down stairs we could improve the stairs detection. Using something like GMM would also allow possible differentiation between walking up and down stairs. This could help improve the floor detection accuracy.

Beacon localization

Localization could also potentially be implemented using Bluetooth beacons. This is done for example at Schiphol Airport. It would be interesting to deploy beacons at known locations and perform a similar localization technique.

NOMENCLATURE

ES	Embedded Systems at TU Delft
GMM	Gaussian Mixture Model
LSQR	Least-Squares Regression
PCA	Principal Component Analysis
PDF	Probability density function
PMF	probability mass function
RSSI	received signal strength indicator
SVM	Support Vector Machine
Wifi	Wireless Fidelity

REFERENCES

[1] R. Alvarez, E. Pulido, and D.A. Sierra. *Climbing/Descending Stairs Detection Using Inertial Sensors and Implementing PCA and a SVM Classifier.* IFMBE Proceedings 60, DOI: 10.1007/978-981-10-4086-3_146, 2016
[2] Min Su Lee, Chan Gook Park, and Chloe W. Shim. *A Movement-Classification Algorithm for Pedestrian using Foot-Mounted IMU.* Proc. 2012 Int. Tech. Meet. Inst. Navig.(Newport Beach, USA):922-927, 2012
[3] Stef Jannsen. *IN4254 Smart Phone Sensing Example 6 code* https://github.com/SmartPhoneSensingDelft/Example6.git 2018