Abstract

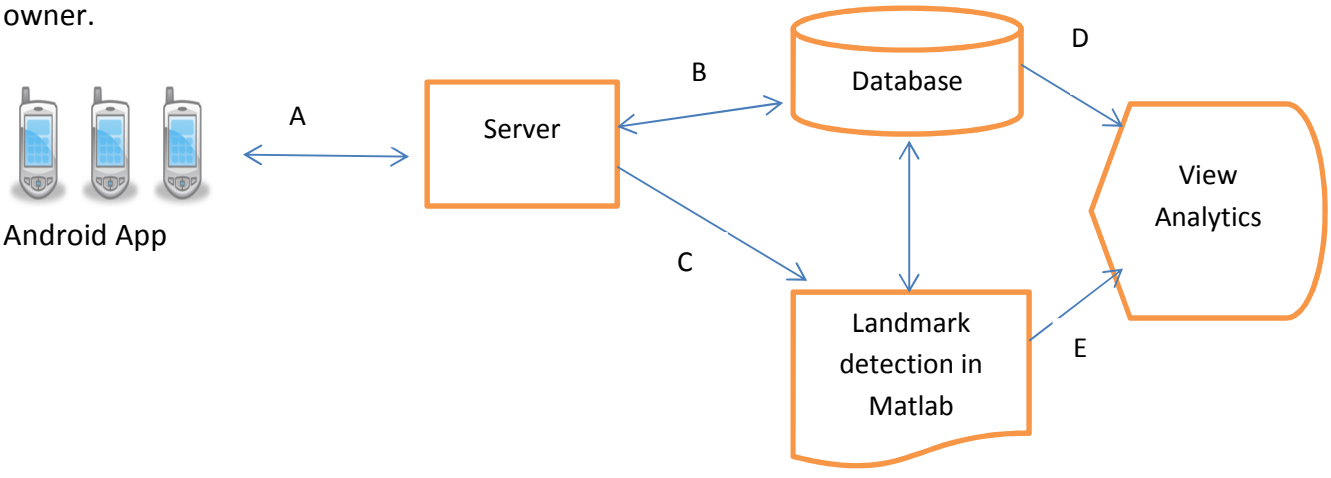
Landmarks are signatures of our surroundings as perceived by the sensors on mobile devices, which help us to uniquely identify a location. I have implemented an unsupervised indoor localization scheme to help prune these landmarks. The scheme explores the possibility of landmarks from sensor features like Accelerometer, Gyroscope, Magnetometer, Light, Sound, WiFi, GSM signal strength. Moreover, it also includes an adaptive clustering algorithm to help us get stable landmarks depending on the varying characteristics of different sensors, day, time and person. Using these landmarks it is possible to identify locations easily. So with the help of these landmarks, an analytics app has been developed for malls which will have the option of annotating these landmarks with comments, pictures and other data.

Updates made during the GSoC Coding period can be found at http://ananthgsoc.blogspot.in/

Architecture and Overview

The project is made up of 4 main modules such as:

- Landmark detection and annotation code in Matlab
- Android Application of the Mall to collect sensor data and provide an interface for users
- Apache Server side code in PHP and database in MySQL to interact with Matlab and the android application.
- Java application on the Server, to provide a view of the analytics generated, to the mall owner.



Working

A. The android App, collects the following data and interacts with the server

a. The annotation of seed landmarks (initial markup) as and when required by the mall owner. This allows him to specify initial information about the building which can act as reference to him when viewing the analytics; ex: name of aisle in the mall, corners, etc.
b. The sensor data collected by the shopper right from the point he enters the mall, till he exits is submitted to the server. Also periodically, some data is sent to the server to ascertain the current position of the phone in the building.
c. Push notifications are sent to the phone containing comments and offers annotated to the nearest landmarks. This is done by a PHP file which is run every 5 minutes as a cron job.

B,C. The server side code written in PHP acts as an interface between the phone application and the Matlab modules.

a. The sensor data files collected is stored in the data folder of the Server
b. The Matlab module is then run with the data folder as input.
c. It generates the stable landmarks, location traces and annotated comments and stores them in the database.

D,E. The analytics viewer goes through the landmarks and location traces of various data collected and shows various analytics that is helpful for the mall owner.

a. Heat Map of the location traces of the various customers of the mall
b. Comments Annotated to points in the physical space and its sentiment analysis
c. The trend in various commodities based on the number of times people have visited a location.

Matlab Modules

The following Matlab scripts act as entry point of code under different scenarios

- parsedata.m
  - This is the module called when the user uploads all the sensor data upon exit of the mall.
  - This calls other modules responsible for Location Tracing, Landmark detection, stabilizing the landmarks and annotation of the comments.
- updateLocation.m
  - This is the module called when periodic updates of the user's location are sent to the server.
  - It comes up with an estimate of the user's location and updates the "landmarked" the user is closest to.
  - It also creates a file containing the notifications the user is supposed to get according to his position. This file is stored in Matlab/stable/notifications_[uid].txt

- enterSeed.m
    - This module is called when the mall owner enters the information about seed landmarks.
    - The estimate of the location of these seed landmarks is calculated.
    - The entered data is stored in Matlab/stable/seeds.mat

The main Matlab modules can be broadly classified into the following categories

- Location Tracing
    - getLocation.m
        - Contains the main logic to do "Urban Dead Reckoning" as explained in the paper
        - Uses the pedometer algorithm and other corrections based on seed landmarks.
        - Takes as input accelerometer, magnetometer, orientation and gyroscope data and outputs the location in x,y co-ordinates.
    - detectCorner.m
        - Detects corners based on gyroscope and orientation signature.
        - Used to detect if seed landmarks are encountered
    - detectElevator.m
        - Detects elevator based on its accelerometer pattern
    - detectVariance.m
        - Used to differentiate between stationary, walking and stairs.
    - correctSeed.m
        - Corrects the location trace based on the detection of seed landmarks.
- Landmark Detection Mechanism
    - getClusters.m
        - The main file which handles all the landmark detection.
        - Sensor Data is filtered, sampled at a uniform rate
        - Clusters are created for different feature combination
    - getLocationClusters.m
        - This contains the main clustering code for detecting a sensor landmark.
        - Clusters which are close in feature space and physical space are termed as landmarks.
    - kMeansInitAndStart.m
        - Uses the optimum k in kMeans
        - Finds the optimum initial seeds for clustering after sampling different subsets.
    - kMeansMod.m
        - Implements the kMeans algorithm for clustering.
        - Optimum k and initial seeds are given as input
    - optKinKmeans.m
        - Finds optimum k in kMeans, using a measure of minimum inter-cluster distance.
        - High Coherence and Minimum coupling for this optimum k.

- Stabilizing the landmarks
    - Stabilize.m
        - This module incorporates the landmarks detected into the database of stable landmarks
    - getStableClusters.m
        - Contains the main logic of when to combine two similar landmarks as mentioned in the paper.
        - Again, further pruning is done to remove outliers and entered into the database and also stored in Matlab/stable/cluster.mat
    - correctLocation.m
        - Correct the location of the current trace based on the landmarks encountered.
        - Angular rotation of the future points lets us provide a more accurate estimate of the trace. Stores the new location in [data folder]/newLocation.mat
- Annotation of Comments
    - annotateComments.m
        - Annotates the comments with the nearest landmark.
        - Enters the comments into the database and also stores in Matlab/stable/comments.mat
- Analytics Generation
    - heatMap.m
        - Goes through the data folders taken as input.
        - Creates the stable landmarks generated from only those traces.
        - Creates an image indicating the distribution of these landmarks. {Red indicates more landmarks, dark blue indicates less landmarks}
        - Creates an image indicating the location Heatmap based on the landmark-corrected location traces. Again, red indicates more heat (foot fall) in that area.
    - readComments.m
        - Reads all the comments currently entered in the system.
        - Generates an image of the comment cloud, along with the location where it is stored.
        - The ratings associated with each comment denote the sentiment associated with each comment. A distribution of this sentiment is also generated in an image file. Here Red means, positive comments, Blue indicates negative comments.
    - getTrend.m
        - Goes through the selected traces and classifies each point in the physical location with the nearest commodity in the mall.
        - This indicates the amount of time spent with each product by the consumer.
        - An average value is depicted in a pie chart.

Server Side PHP Code

The server side code can be broadly classified as follows:

- File Upload handling
  - fileUpload.php
    - Uploads the files sent by the phone to a new/existing folder, as the situation demands.
    - All the files of a specific trace are stored in a single folder in the Server/www/data/ directory.
    - It calls the required matlab module.
- User Login and Registration handling
  - login_api/index.php
    - Handles User registration and login.
    - Also handles the device registration required for Google Cloud Messaging.
- MySQL database interface
  - login_api/include/DB_Connect.php
    - Connects to the MySql server on the machine
  - login_api/include/DB_Functions.php
    - Implements certain functions like user verification, registration and executing a query
  - login_api/include/config.php
    - Contains the user information for MySQL server and also the Google API key required to use the Google Cloud Messaging.
- Send Push Notifications
  - login_api/GCM.php
    - Implements the functions required to send a message through Google Cloud Messaging service.
  - login_api/send_notification.php
    - This is used as an executable and not through some web service.
    - It is run as a cron job, every 5 minutes.
    - It checks if some user's location has been updated which makes him eligible for receiving the comments/offers near him.
    - Sends the message to each such user.

Android Application

The Android Application can be broadly classified into

- Sensor Data Collection
  - SensoSaurActivity.java
    - Contains the main code which binds the sensors of the phone to their listeners

        - Contains logic to log data such as comments, seed landmarks, etc in their respective files.
        - Passes on the control for uploading the files to the server.
    - SensorInfoSelect.java
        - Selects the sensors required for logging
        - Configures for a specific resolution.
- UI
    - SensoSaurActivity.java -> main.xml
        - Main UI to log data [default view – logged in]
    - LoginActivity.java -> login.xml
        - User Login UI and its handlers[default view- logged out]
    - RegisterActivity.java -> register.xml
        - New User registration and its handlers
- File Upload
    - ServerCommunication.java
        - Performs files upload in background as an Async Task
        - Uses http apache API
    - LoggingActivity.java
        - Schedules the periodic logging of Accelerometer, Magnetometer, Gyroscope and orientation Sensor, once every minute
    - JSONParser.java
        - Reads the response from the server in JSON format.

- Google Cloud Messaging Registration
    - GCMIntentService.java
        - Handles the receiving of Push notification messages
        - Generation of Push notification on the phone by enabling wake lock.
- Database Handling and other utilities
    - DatabaseHandler.java
        - Implements SQLite database to store the user's credentials.
    - UserFunctions.java
        Helper functions for login and registration
    - CommonUtilities.java
        - Notification Generation helper.

Java Analytics Viewer

The Java Analytics Viewer is made up of the following modules

- MainActivity.java
    - Contains the handlers of various button click events in the UI

    - Calls suitable functions in the “visualize” class present in Sensosaur.jar [This is a package of the required functions to generate the analytics]
    - Displays the relevant images in the tabs
    - Displays the comments in a tabular format for reference.
- Sensosaur.jar
    - This is the output after compiling the Matlab files as a Java Project.
    - It is packaged in com.Sensosaur.*. All the Matlab files are available through the “visualize” class.
- javabuilder.jar
    - This is available in the Matlab Compiler Runtime installation directory. Used to run Matlab Builder JA compiled files.

Database and Data storage

- The database “landmark” has the following tables in MySQL
    - Users
        - Contains user info like username, password, salt, phone, GCM_registration_ID, last time when location updated, last time when notification sent.
    - Sample
        - Contains the information regarding each location trace. The folder in which in it is stored, user who created it and when it was created.
    - Landmark
        - Contains the information about landmark, its data folder, the sensor feature number, centroid of landmark, number of points.
        - Confidence count indicates the number of times that landmark has been encountered by the user
    - Comments
        - Comments contain the rating, comment and the landmark ID it is annotated with.
- Apart from the database, the corpus of stable landmarks and seeds are stored as Matlab object files for faster access. They can be found in the Matlab/stable subdirectory
    - Cluster.mat
        - Cell Array containing the stable clusters and its stable Features
    - Comments.mat
        - Cell Array containing the comments and their information.
    - Seeds.mat
        - Seed Landmarks entered by the mall owner along with their estimation
    - Areas.txt
        - The commodities sold corresponding to the seed landmark entered by the mall owner. This currently has to be entered manually.

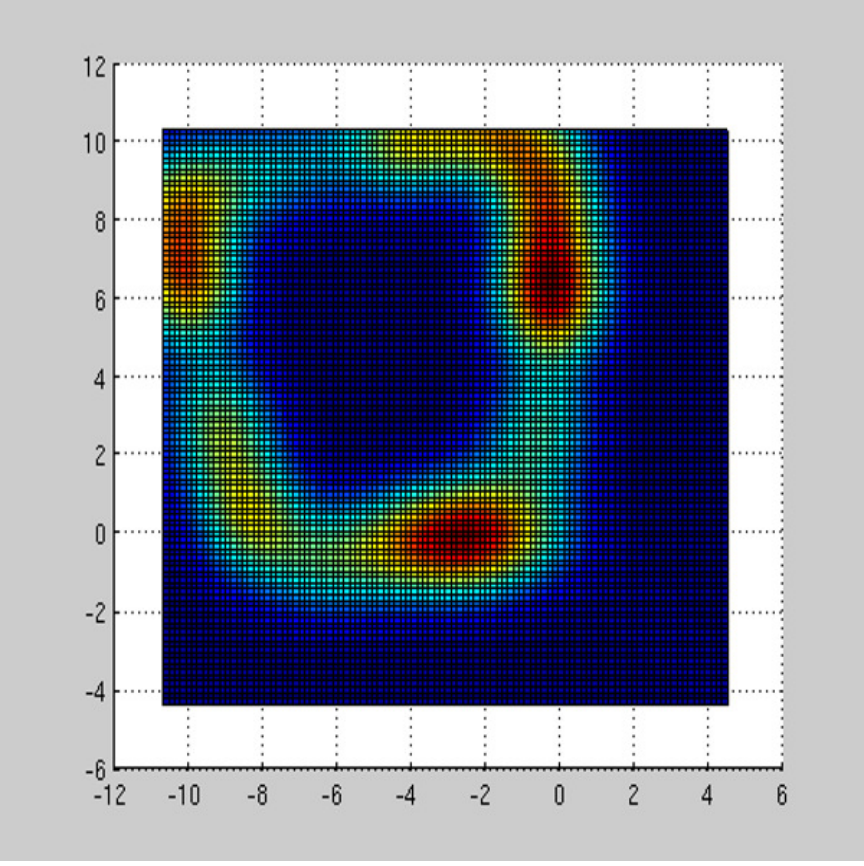
Experiment

The experiment was conducted by simulating an actual market scenario in the engineering building of my college. The mall owner first annotated the seed landmarks using the “Annotate” mode in the Phone Application. He marked the various aisles of the market, which was later cross referenced to the commodities available in those aisles present in “Areas.txt”. This way we get an idea of the crude map of the mall and where the commodities are located. This was repeated multiple times and the averaged result is taken as the initial seed landmarks.

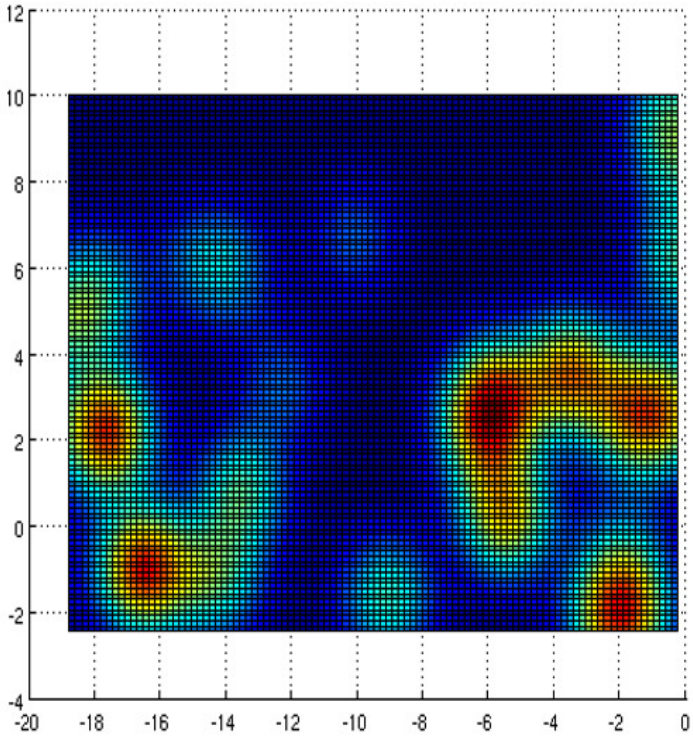
Now, as customers, two people used the application in two different phones (Nexus 4 and Galaxy S3). They started from the same position and roamed around the building, commenting and “ticking” (which indicates increased activity, like a virtual buy in that aisle). The sensor data of the entire trace was sent to the server by “Log In” the activity. It was observed that as the users roamed around, they got the comments annotated with the nearest landmarks as Push notifications. Note that the offers given by the mall owner can be entered as a comment by the mall owner. When the user exits, he stops the logging and exits.

Results

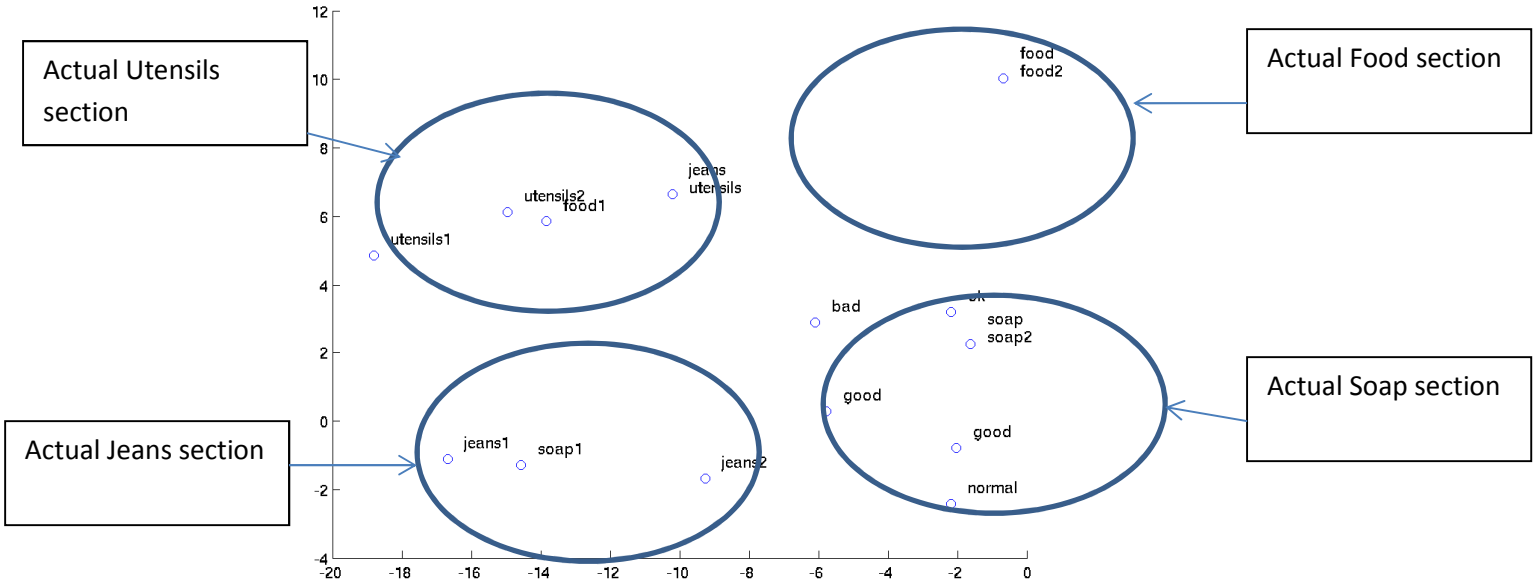
- Heat Map of the location traces



- Density of stable landmarks

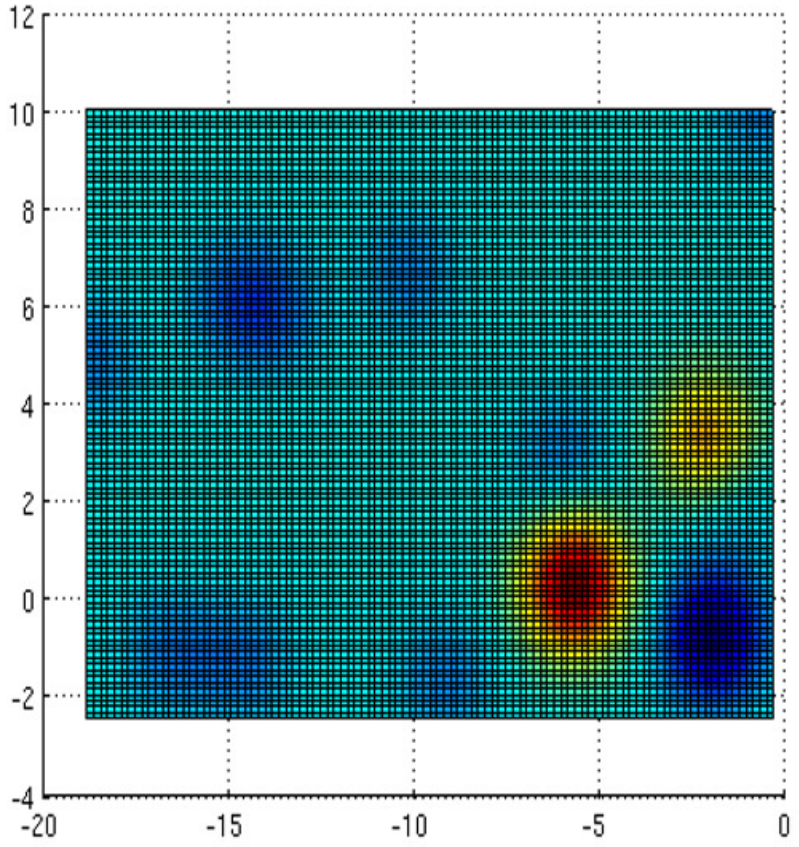


- Comment Cloud



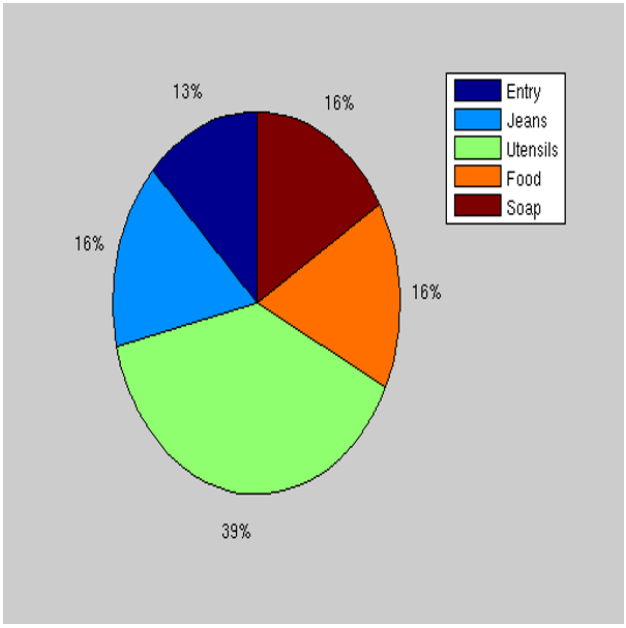
As can be seen, while most of the comments are annotated correctly, there are some outliers with the neighboring aisle sections.

- Comment Sentiment analysis {Red denotes positive rating}

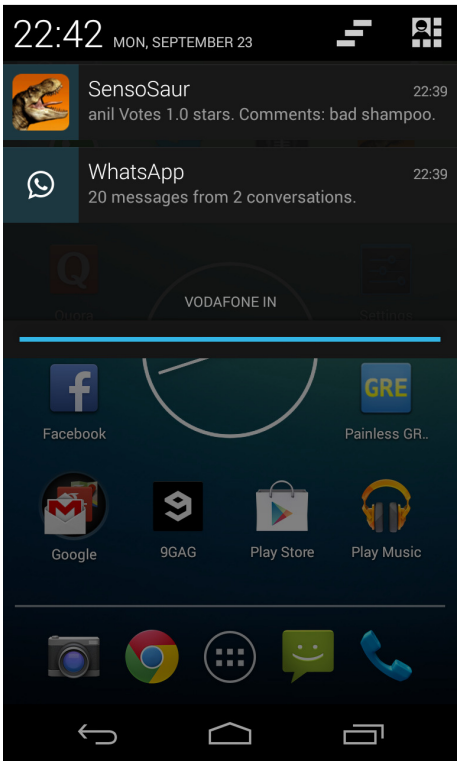


is

- Commodity Trend



- Push Notification



Future Work

- Some cases where the entire chunk of files could not be transferred.
    - Cause: Slow Mobile Data connection
- Size of files sent to server to update location can be minimized.
    - Solution: Pausing and resuming logging sensor data
- Overlay on map of mall to be automated
    - Requires proper correction of seed landmarks
- More accurate activity recognition to detect stationary, stairs, walking
    - Currently, the error is considerable mainly due to inability to properly classify current activity
- Experiments carried out with constant orientation of the phone
    - Testing should be done with other orientations and suitable adaptations.
- Periodic Pruning of unstable landmarks
    - Currently the number of landmarks explode, even if a manageable number of them are stable
- Comment annotated with only one landmark
    - This might lead to a sparse cloud of comments as landmarks increase with more data samples.
    - Solution: Annotate comment to a bunch of landmarks, instead of just one.
    - This increases the amount of data to be pushed to the phone

- UI for the mall owner to enter offers and aisle information
    - Sufficient information about the aisles can be currently entered only in the form of comments
    - A special interface to enter offers can be thought of, so that offers are annotated to a large area, and not just a landmark
- Handling of multiple data requests on the server side has not been tested.
    - Suitable modifications in data storage and threading might be required

References

1. Original landmark idea: http://people.ee.duke.edu/~hw98/unloc/
2. The author is a co-author of a paper which explains the idea behind this analytics. The paper is found in the documentation directory.