Homework 4 (Due: before class in week 10)

Part A. Place Tracking
In HW 3 you looked at low-level mobile computing—sensors inputs and platform energy considerations. In this homework, you will explore high-level mobile computing considerations that are pervasive to apps. Specifically, you will map low-level location samples to high-level places—people don’t care about lat/long values, they care about places that are meaningful to them. For motivation, imagine you are developing a journal that lists the places you have been, automatically populated by your mobile based on location sensing. This would be a cool app to develop! It and most other applications that deal with location must grapple with the issue of inferring places with some semantic meaning to the user from raw location data. It sounds simple, but is often difficult due to ambiguities as to what makes a natural concept of place for users along with variability in the sensor data. In this assignment, you will design and test an inference scheme to see for yourself.

You are encouraged to work with a partner (teams of two only) for this part of the homework, and to help each other with Android questions (but not actual homework solutions) on the discussion board.

Experimental Setup
You will use your Nexus One mobile for this experiment as before. The baseline app that we provide you runs continuously to gather regular location samples, and interprets those location samples as places. It does not do a very good job of interpreting location samples as places. Your task is to develop your own functions that do a better job of mapping location to place. To do this you will have to gather some location data of your own as well as ground truth for the places you visited (which you know by definition) and experiment to see how well your inference procedure works.

Note that in HW 3 you considered the energy cost of gathering location samples. While a real app would care, you should not worry about that for this homework, as we will not care for grading. Accordingly, our baseline uses both GPS and WiFi to get the best location at all times regardless of energy costs.

Place versus Location
We give you only an intuitive definition of place; defining it carefully in terms of location samples is part of the assignment (and what any app developer must deal with). By “place” we simply mean something that users would consider to be a fixed location, such as “work”, “home”, “the Starbucks on 35th”, etc., but not “your car” since that moves. However, in your homework you do not need to associate labels with places. Thus you might refer to “work”, “home” and “the Starbucks on 35th” as places #1, #2, and #3. Ideally, an application should label these places, and you may consider how to do this and write an UI if you would like.

Several aspects of place are key to this assignment. First, different location values may correspond to the same place. For example, suppose that you record the location value at a place, leave, and then return to the same place and record the location again. The two location values are very likely to be different, yet the place is the same. Second, when you are moving between places you are not at any well-defined place. You are in transit. At least two kinds of transit can be recognized: transit by foot (when
you are walking, jogging, etc.) and transit by vehicle (when you are in a car or bus). Third, places change smoothly over a human timescale. You might get in your car and drive from home to that Starbucks, and a place journal might say “home”, transit(car), “Starbucks”. It would not be reasonable for a place tracking app to say “home”, transit(foot), “home”, transit(car), “corner of 35th”, transit(car), “Starbucks” if you happened to reverse into your driveway (thereby momentarily coming closer to home) and were held up at a long traffic light near 35th street. Fourth, places are large enough to be descriptive of a location you would tell someone to meet you, but not overly large. For example, you stay at the same place as you wander around your home, but go to a different place when you visit your neighbor’s house. There may well be other aspects of place we have not discussed, but that should be enough to get you started.

Your goal is to convert a time sequence of location samples into a time sequence of inferred places and transits by either foot or vehicle between them. The sequence of places and transits will tell you where you have been during the day (or experimental period) as well as how long you were at each place and how long it took you to travel between places.

**PlaceTracker Application**

As a starting point, we give you the PlaceTracker application that gathers location data as often as possible from all available location providers. The application will run continuously unless you explicitly stop it. There are three functions of interest:

1) `GotLocation()` which is called at every location update
2) `InMotion()` which determines if you are in transit or stationary and
3) `AtPlace()` which determines if you are stationary at a new place or if you have been at the place before.

The latter two functions currently have very naïve implementations that you will need to refine. There is also a LogLocation function we would like you to use to log data. It logs the time, lat/lon, measurement accuracy, Place and a Boolean inMotion. Place is a string that you provide when stationary which describes a place (though the string can be as simple as “Place-1”). When inMotion is True, Place is automatically set to “NA”.

**Goal and Development Steps**

Our PlaceTracker interprets location samples in a naïve way. Your goal is to do a better job! We suggest that you proceed as follows:

1. Develop and test logic to display whether you are stationary at a place or moving between places. With this logic you will be able to divide a trace into a sequence of place and transit events.
2. Develop and test logic to recognize places you have already visited as the same place.
3. Develop and test logic to distinguish transit events into transit by foot and transit by vehicle.

By working on these individual tasks and making your logic robust across multiple runs you will make progress towards the goal. As you test each step, try it on cases for which you have obvious ground truth – do not test it on tricky corner cases, at least until you have a robust version.
PlaceTracker Output: The version of PlaceTracker that we give you outputs a log message at every location update. Your version of the application should generate just one message for every Place you visit and one for every time you are in transit. These messages should indicate how long you were stationary or in transit, and for transit periods they should indicate if you were moving by foot or by vehicle. You can either modify the current LogLocation function to do this, or write these messages to a second log file (useful for incremental development!). This form of output is what you will hand in.

Replaying GPS Traces
The Android emulator allows you to “replay” a GPS trace, described here:
http://developer.android.com/guide/topics/location/obtaining-user-location.html#MockData

This will be helpful for developing your application, and you will need to use it for part of the assignment (see Turn-in below). To play back a full GPS track, you will need to record the track in GPX or KML format. MyTracks is an app available from the Market that will record tracks and can export to either format.

Note: There is a bug in the emulator where if you target the Android 2.3.3 SDK, GPS playback will crash the emulator. A workaround is to use the 4.0 SDK for the emulator. You will still target the 2.3.3 SDK for deploying the application to your device.

Note: It is only possible to relay GPS locations, not WiFi locations, and the GPS locations have no associated accuracy. Thus your app must be able to work with only GPS locations for evaluation purposes, even if you make use of additional location information when you are running “live”.

Turn-in
Turn in the following items:
  a) Inference Description. Write a brief description of your algorithms that determine transit vs stationary periods, whether you have visited this place before, and whether you are moving by foot or vehicle. (Target: 1/2 page)
  b) Inference Experience. Describe two ways in which you improved your algorithms based on your experience or experimentation that were not obvious to you initially.
  c) Source. Provide the source code such that we can inspect, compile and run it as needed.
  d) Sample output for your trace. Provide the place tracking output for one sample run for which you know ground truth, annotated with your comments about how well the inference worked compared to ground truth. Try to pick something interesting to show us, perhaps a fairly tough case on which you do well.
  e) Sample output for our trace. We have provided you with a reference location sample trace that you may replay to your application using the GPS emulator. Run your app against our trace and turn in the log that records the places that were visited (and the time spent at each place) and the instances and duration of transits and the mode of transit.
Having Fun
You might be inclined to develop your app by running it against our reference location trace until the output looks reasonable. Avoid this tendency! We think that you will find it an enjoyable challenge to really make place tracking work well, and we’d like you to have a bit of fun with the problem. To do that you’ll need to try it for cases in which you know the ground truth. You’ll need some decent scenarios for which you can get locations, for which you might leverage your efforts on HW3 (or go with a SIM card). And you’ll need to repeat experiments to see how fragile or robust it is.

To help us all have some fun, you are free to discuss place tracking issues and how well your algorithms are working with other students in the class, e.g., using the discussion board, and may even try out other teams apps to compare them against your own. All we ask is that only you and your partner should inspect and develop your own code.

Part B. Textbook.
6.12, 6.14, 6.15, 6.29, 6.30, 6.32