Campus Assistant Application on an Android Platform

Mihaela Cardei, Iana Zankina, Ionut Cardei, and Daniel Raviv
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, FL 33431, USA
E-mail: {mihaela@cse., izankina@, icardei@cse., ravivd@}fau.edu

***Abstract*—College campuses can be large, confusing, and intimidating for new students and visitors. Finding the campus may be easy using a GPS unit or Google Maps directions, but this changes when you are actually on the campus. There is no service that provides directional assistance within the campus itself. This paper presents the architecture and design specifications for a campus assistant application on an Android platform. Scenarios are illustrated for the Florida Atlantic University - Boca Raton campus.**

Keywords: campus assistant, Android platform, campus map editor, ubiquitous computing, Google map.

I. Introduction and Related Works

Recent technological advancements have gain popularity finding many applications in everyday activities. There are many devices and different applications that focus on directing the user to desired locations. Today's drivers are well equipped for travel thanks to the GPS units [4] many have in their cars. GPS applications allow users to enter a destination and based on their current coordinates display the fastest way to the destination [8]. Additional features have evolved over time, such as displaying congested routes, which allow users to make smart driving decisions and improve driving safety. This saves time and stress when going to unfamiliar places or taking long trips.

Google Maps and GPS systems have become indispensable in recent years, with vast amounts of users relying on them for directions [6], but their capabilities have not yet been fully applied to university campuses. Directions within campuses are not available using the Google maps application. Taking for example the Boca Raton campus at Florida Atlantic University (FAU), when the Engineering East Building, which is the home of the Computer and Electrical Engineering and Computer Science Department, is entered as the Destination no complete directions are provided. The directions lead to the FAU campus, but not to the building. In addition, not all the buildings and parking lots are shown on Google Maps.

Campuses can be quite large and confusing. Visitors, new students, and staff can have a difficult time getting around. Even when they are asking for directions, they often time cannot find the destination because the directions involve knowing the surrounding buildings and landmarks of the campus. In addition, some of the campuses, such as the Boca Raton campus at FAU, are growing fast, with new buildings and facilities being added. This can get quite stressful, especially considering people are often on a schedule and need to get to a certain location on time. Eliminating this stress and confusion would improve the overall atmosphere on campus. Since smartphones are a ubiquitous technology nowadays, it makes sense to use them to facilitate campus directions.

The problem that we address in this paper is on using the current advances in technology to provide a mechanism to facilitate users navigation in campuses. Our objective is to design and implement a user-friendly system that provides driving directions to buildings and parking lots. The system provides both driving and walking directions to the destination. The user has the option to select a parking lot close to the destination building, based on the user type (e.g. visitor, staff, or student).

One of the most popular smartphone platforms is Android [1], which is a Linux-based operating system designed primarily for touchscreen mobile devices. Google has released its code as open source, triggering a large community of developers to write applications that extend the functionality of the device using a customized version of Java. Due to its superior technological capabilities, we propose to develop the campus assistant application on the Android platform.

There are some initiatives related to our project. Researchers from The University of California at Santa Barbara are working on an Interactive Map Project [7]. Their map allows users to select buildings to zoom into and locate, as well as finding a room within a building. This project does not support navigation.

Another application is the University of California,

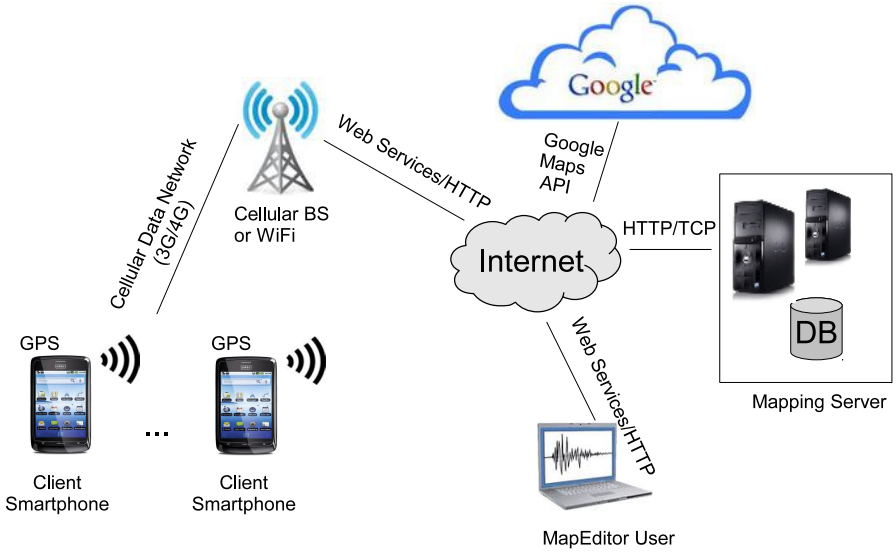


Fig. 1. Campus Map Architecture

San Diego campus map. This map is accessed through a web browser and allows the user to select a source location and a destination. When this information is submitted, the shortest path is outlined on the map, and the distance and expected walking time is shown [5]. This web-based application does not allow however for driving directions within the campus.

There are also several apps [2] that provide campus maps for a number of universities and offer services such as a directory of buildings and locations, building photograph gallery, display of current location, zoom in/out. None of these applications however implement both driving and walking directions, navigation, considering the user type.

II. Campus Assistant Application Architecture

In this section we present the architecture for the campus assistant application. We derive the following requirements for the system:

- The application is available on the Android platform.
- The application allows a user to select a source and destination locations and displays the shortest path. The user can select whether this is a walkable or drivable path. This feature is especially useful on campus, since people often walk between buildings. Alleys and various shortcuts can be used in this case.
- The application provides navigation capabilities based on the user type. A user enters the destination location and based on the current location a shortest driving or pedestrian path is displayed. If the destination is a building, then based on the user type (e.g. student, staff, visitor) a path to a compatible parking lot is provided. This is particularly important in campuses, where there are different parking lots assigned to visitors, students, and staff.
- The application provides rerouting if the user departs from the projected path.

Figure 1 presents the architecture of our campus assistant application. Since Google Maps does not provide any information on campus locations (e.g. buildings, parking lots), we have to build the map data structures that describe campus locations, roads, alleys, traffic signs, everything needed for directions and navigation. We designed a MapEditor tool to edit and manage campus maps. These maps are stored on the server as XML files.

The MapEditor tool uses HTML 5, JavaScript, and jQuery. Google Maps API v3 is used to display the map tiles and the various markers representing campus map concepts (buildings, etc.). This web-based tool is stored on the **Mapping Server** and can be accessed from smartphones and PCs using popular web browser such as Opera, FireFox, Internet Explorer, and Chrome.

The MapEditor tool allows fast and easy manipulation of a map and its XML file. Since buildings, streets and other aspects of the campus are not searchable on Google Maps, this has to be done manually by the **MapEditor User**. The MapEditor lets the user edit a campus map superimposed on the Google Map and saves it as an XML file on the server. It also has version management for existing XML files and also allows users to directly manipulate the map XML representation.

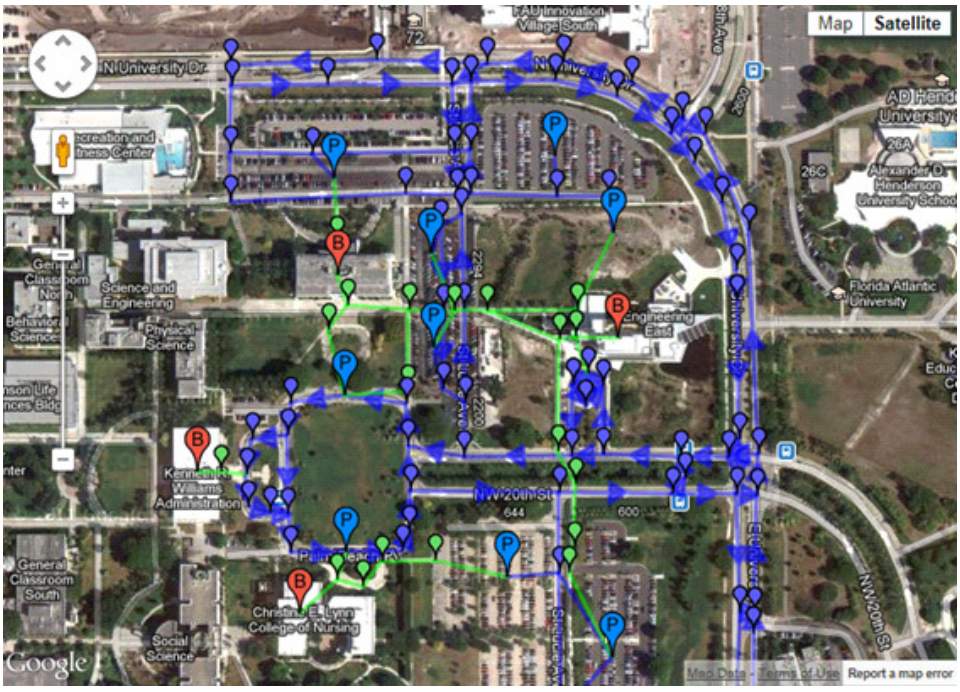


Fig. 2. FAU Campus Map Opened in the MapEditor

When the campus assistant app is deployed, the MapEditor is used to keep the campus map up to date. Smartphone users automatically download the most recent version of a map and this simplifies maintenance. Map editing is useful when new locations (e.g. buildings) are being added to the campus or locations change their names.

The following features are available to the **MapEditor User**: load an existing campus map, edit map features (create/edit buildings, parking lots, road/alley segments, traffic signs), save a map to the Mapping Server as an XML file, display the map XML representation. Figure 2 shows a screenshot with a section of the FAU campus map loaded in the MapEditor, with icons for buildings ("B" markers), parking lots ("P" markers), walkable (green), and drivable (blue) segments. These map features will be discussed in section III.

The main component of our architecture is the **Campus Assistant smartphone app** that runs on Android phones. The app user interface prompts the user to enter relevant information such as user type, campus map, and destination location. The app provides both directions and walking/driving navigation capabilities while on campus. The app relies on a JSON/HTTP protocol to request an XML map file from the **Mapping Server**. This XML file is parsed and the map graph features are constructed. The smartphone GPS device is used to determine the user's current location or the user can indicate a source location. If the user wants driving directions, the app will direct the user to the compatible parking lot nearest to the intended destination.

The app then computes the shortest path between source and destination, and displays it on the user's screen on top of google maps. Icons identify the user's current location and the destination. More details will be discussed in section IV.

III. The Map Schema and the MapEditor

We represent a map as a graph where vertices describe map features (locations such as buildings, parking lots, and intermediary segment ends) and edges describe walkable and drivable segments between vertices: alley or road segments. A map is stored as an XML file on the Mapping Server. The XML schema defines the hierarchy of objects and attributes from a map graph. We present this hierarchy in a UML class diagram in Figure 3.

The root of the class hierarchy is the *CampusMapObject* class from which all other objects inherit, except the *LatLng* class. The root class provides every descendant with an individual id attribute. This id is used for referring to map graph elements.

The *Vertex* class represents a location (map feature) at the most basic level. A vertex has an attribute called names that is an array of strings to store the names of that particular location. Buildings and parking lots have an official designation (e.g. Engineering East Building) and several other common names, such as "EE96" or "Green Building", "Engineering Green Building", "New

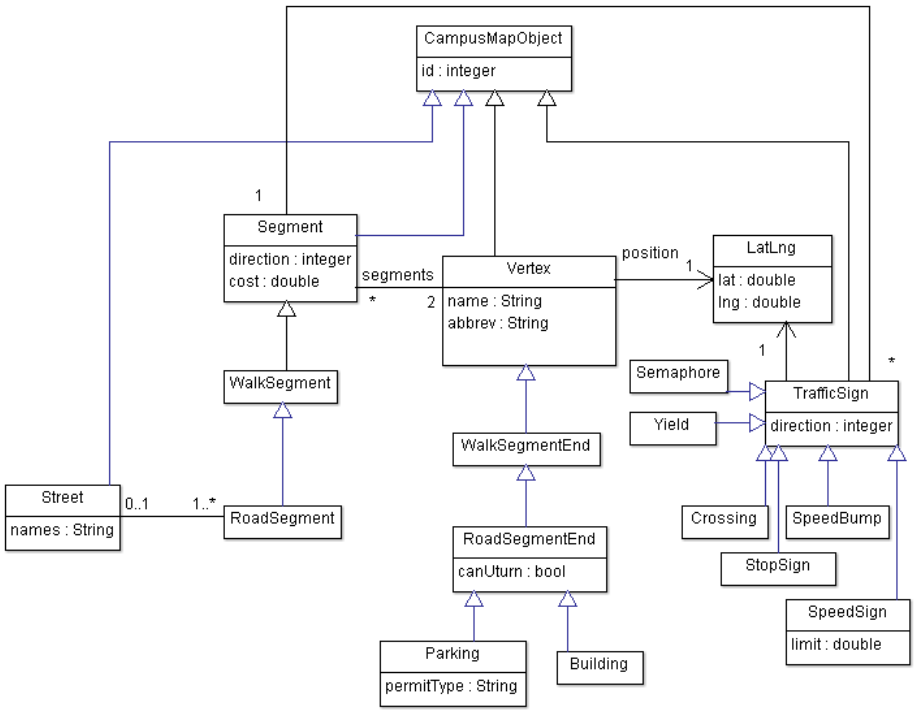


Fig. 3. UML Campus Map Class Diagram

Engineering Building", commonly used by the academic community. The abbrev vertex attribute is used to display a label on the mapping user interface and usually is set to the official location abbreviation, such as "EE96" for the Engineering East building. A vertex indicates its location on the map using the *LatLng* object that stores the latitude and longitude as double numbers.

Vertices are specialized on whether they are reachable by car or not. A *WalkSegmentEnd* object describes a location that is not reachable by car, but only by walking. This type of object is used in large numbers to build multi-segment alley paths that connect map features and are accessible only to pedestrians.

A *RoadSegmentEnd* object represents locations that are directly reachable by car. Parking lots, buildings (in general), and vertices used to model roads in multi-segment paths are *RoadSegmentEnds*. We consider that a location accessible by car is also accessible by foot, so we made the *RoadSegmentEnd* subclass of *WalkSegmentEnd*. This object has a boolean attribute canUturn that indicates whether it is possible to make a U-turn at that vertex.

The *Building* class models campus buildings, including lecture halls, residence halls, facility plants, gyms, and the stadium. The *Parking* class holds the information for a parking lot on campus – the permit type in addition to its abbreviation, and list of names. Our university issues separate permit types for faculty and staff, students, and visitors. The Campus Assistant app can find directions to the nearest parking lot that can be used by the user with the corresponding permit type.

A *Segment* object connects two map graph vertices and represents an edge in the map graph. A *Segment* object stores its cost computed from the distance on the map between its two endpoints expressed in meters. The direction attribute indicates whether a segment is navigable in one direction, $end_1 \rightarrow end_2$, in reverse, or in both directions. Direction for a segment is meaningful also for WalkSegmentEnds such as escalators, so this attribute was factored out to the *Segment* base class.

The *WalkSegment* class inherits from *Segment* and, as expected, represents a segment between two locations that can be navigated only by foot, not by car. The *RoadSegment* class describes a segment that can be navigated by car. A *RoadSegment* is associated with one *Street* object or none, as some segments in parking lots or next to buildings don't belong typically to a street. A *Street* object can have multiple names, as it is common in the USA for county and state highways.

We represent traffic signs in our schema with the *TrafficSign* class. A *TrafficSign* object has a *LatLng* location and a direction attribute indicating the driving direction in which the sign applies, relative to the *RoadSegment*'s $end_1 \rightarrow end_2$ direction.

The schema supports a number of classes inheriting from *TrafficSign*: *Crossing* (direction=0), *Stop Sign* (direction=$\pm$1), *Semaphore* (direction=0), *Speed Bump* (direction=0), *SpeedLimit* (direction=$\pm$1), and *Yield* (direction=0). These classes are used to augment the user experience in the navigation app with audible and visual indicators when the user approaches a traffic sign and gives no sign that he/she will slow down.

The MapEditor is a tool we developed to allow authorized users to edit features on a campus map. We expect the tool to be used initially to create the map for a campus, enhanced with features needed for driving and walking navigation. After that, the tool is used occasionally when updates are necessary, such as when a new building is inaugurated, a feature changes its name, a new traffic sign is added, and so on.

The MapEditor user can add new objects and edit their properties by interacting with the objects' view displayed superimposed as markers and lines on a google map. For example, a left click on the map places a new vertex; a click on a vertex toggles selection state; right-click on a map marker or on a segment overlay edits its properties. With a vertex created, the user can create a new segment to another vertex by left-clicking it. A left-click on a RoadSegment toggles its direction attribute; a right-click on any segment deletes it, and so on.

The MapEditor tool is implemented in HTML 5 and JavaScript. The Google Maps API v3 provides convenient map display in a browser, zoom-in/out, and an API for manipulating latitude/longitude and for computing geographic distances. The MapEditor uses a JSON based protocol to send/receive map XML files and map metadata to/from the Mapping Server.

Figure 2 shows the FAU campus map open for editing in the MapEditor, with markers representing vertices and colored lines representing segments.

IV. Campus Assistant Application for the Android Platform

The Campus Assistant app offers directions and walking/driving navigation on one's Android smartphone. The steps performed by the app are as follows:

- Prompts the user for information: the user type (student, staff, or visitor), the campus (e.g. Boca Raton, Davie), and the destination location – usually a building or a parking lot.
- Downloads the campus' map XML file from the Mapping Server using HTTP.
- Parses the campus map XML file and builds the campus map graph data structure.
- Calculates the shortest path from the source to the destination (detailed below). The current GPS location is used as the source. The destination is computed based on the user type. For example, if a driving path to a building is selected, then driving directions to the closest parking lot where the user is permitted to park are provided.
- Directions and navigation prompts are displayed on the campus map on the Android smartphone. The shortest path is displayed, overlaid on the google map view. The current user location is displayed using a car/pedestrian icon. In addition, the user has the following options: select drivable or walkable path, select a new destination using the phone's touchscreen by tapping on the destination vertex marker, and zoom in/out with pinch gestures.

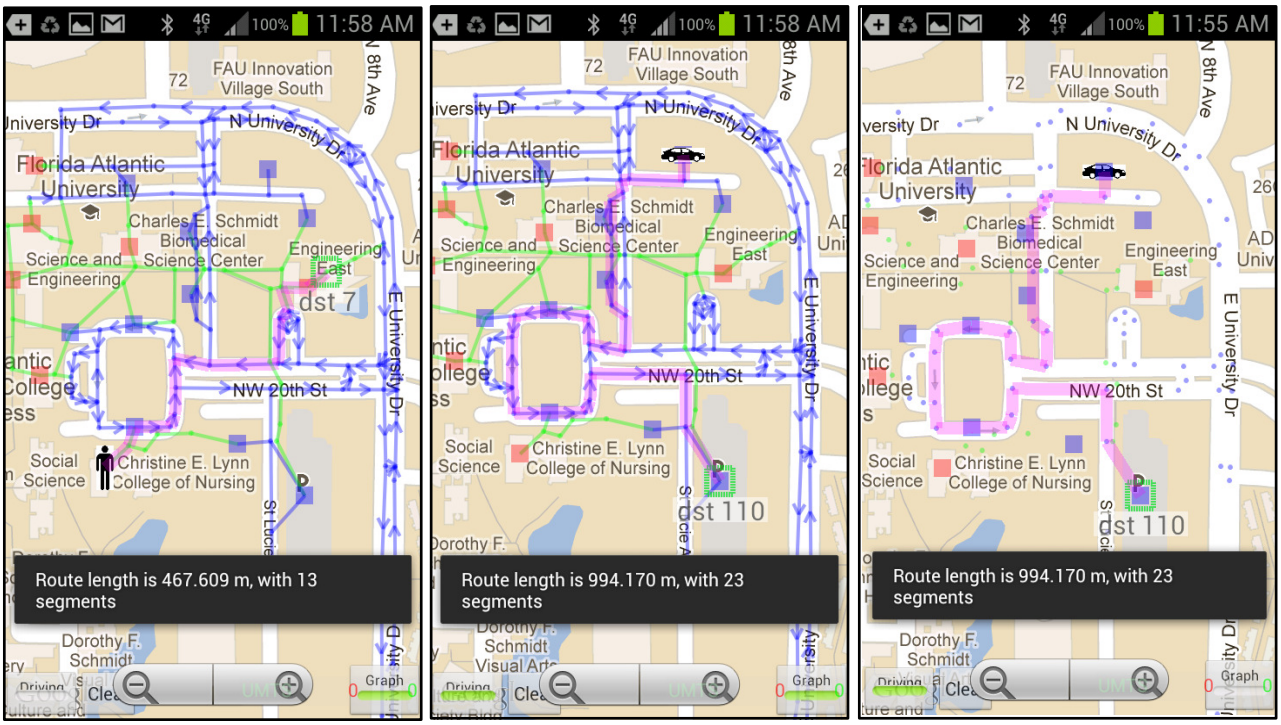
We implemented the app in Java using the Android SDK. For this project we used the Eclipse IDE and utilized the ADT plugin to edit files and manage the project. The Android API provides many useful packages that allow users to access the vast functionality of the Android device, such as the classes for the user interface and for accessing the compass and the GPS device. The Eclipse environment has excellent integration with the Android platform and allows users to debug applications running directly on the phone.

An important step in our application is parsing the XML map file to create the map graph data structures. A parser basically reads the XML document, identifies the tags, and extracts the data between the tags. This allows a computer program to access and use the data from the XML file. We used Document Object Model (DOM) to perform parsing. As result of parsing, we build data structures storing the campus map objects and their attributes, such as vertices (e.g. Buildings, Parking lots, etc.) and segments (e.g. road segments, walking segments).

We compute the shortest-path between source and destination using Dijkstra's algorithm [3]. The original algorithm presented in [3] computes shortest-paths from a source vertex to all other vertices in the graph. The weight a path is computed as the sum of the cost of all segments on the path. The algorithm stores the vertices to which the shortest-path has not been computed yet in a minimum-priority queue. At each step, one vertex with the minimum weight is removed, and its shortest-path calculation is completed.

Since we are interested to compute the shortest-path to a single destination vertex, in order to reduce the runtime complexity, we keep only the next step candidate vertices in the minimum-priority queue. In addition, our algorithm stops as soon as the shortest-path to the destination is computed. The shortest-path is then displayed on the user interface overlayed on top of the campus Google Map in map or satellite view.

In our case, the weight of a segment is the Euclidean distance, which is one of the attributes (e.g. denoted as the cost) of a driving/walk segment. We plan to use more



a. Walking Path b. Driving Path c. Driving Path without a Graph

Fig. 4. Campus Assistant App

sophisticated cost functions for driving directions that consider the estimated driving time based on the number of turns, traffic signs on the path, and speed limit. This cost function can be further enhanced by real-time traffic updates from the University Police department, in case of sport events, construction work, etc.

The user has the option to choose between walking and driving paths. For driving paths, only road segments are considered for the shortest path computation, while for a walking path both road segments and walk segments are presented as candidates for edge relaxation in Dijkstra's algorithm. The complexity of Dijkstra's algorithm [3] using minimum-priority queue implementation is $O(E \cdot lgV)$. For the test map in the screenshot with 330 vertices, the runtime for the shortest path algorithm was 3 ms measured on a Samsung Galaxy S Blaze phone.

Figure 4 shows some scenarios of running our application on the same type of phone. Figure 4 a. shows the walking path between the College of Nursing Building and the Engineering East Building. Figures 4 b. and c. show driving directions with two options, when the underneath graph is displayed or not.

V. Conclusions

In this paper we presented a campus assistant application developed on an Android platform. The application provides walking/driving directions and navigation services to users at Florida Atlantic University campus. We also designed a campus editor that allows authorized users to add new campus maps or update existent ones. Currently we are working on adding new features to our application, such as speech directions and voice activated navigation control.

References

[1] Android-Discover Android, http://www.android.com/about/, last accessed Dec. 2012.
[2] Campus Mapping Applications, https://play.google.com/store/search?q=campus+map&c=apps, last accessed Dec. 2012.
[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, Third Edition, MIT Press, 2009.
[4] GPS.gov Roads and Highways, http://www.gps.gov/applications/roads/, last accessed Dec. 2012.
[5] M. Kelly and D. Lindquist, Campus Map, http://campusmap.michaelkelly.org/map, last accessed Dec. 2012.
[6] J. Kincaid, Marissa Mayer: 40% Of Google Maps Usage Is Mobile (And There Are 150 Million Mobile Users), http://techcrunch.com/2011/03/11/marissa-mayer-40-of-google-maps-usage-is-mobile-and-there-are-150-million-mobile-users/, last accessed Dec. 2012.
[7] UCSB Interactive Campus Map, http://code.google.com/p/ucsb-icm/, last accessed Dec. 2012.
[8] T. D. Wood, How to Use a Hand Held GPS Receiver, Aug. 2012, http://www.rei.com/learn/expert-advice/gps-receiver-howto.html, last accessed Dec. 2012.