Stony Brook University

Location Based Activity Recognition using Mobile Phones

A Thesis Presented

by

Naveen Vignesh Ramaraj

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Science

Stony Brook University

May 2009

Stony Brook University

The Graduate School

Naveen Vignesh Ramaraj

We, the thesis committee for the above candidate for the
Master of Science degree, hereby recommend
acceptance of this thesis.

Dr. Samir R. Das - Thesis Advisor
Associate Professor, Department of Computer Science

Dr. I.V. Ramakrishnan – Chairperson of Defense
Professor, Department of Computer Science

Dr. Jennifer L. Wong - Thesis Committee Member
Assistant Professor, Department of Computer Science

This thesis is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Thesis

Location Based Activity Recognition using Mobile Phones

by

Naveen Vignesh Ramaraj

Master of Science

in

Computer Science

Stony Brook University

2009

Location based services have become extremely popular in the recent years in a variety of contexts. In this thesis work we present a framework that uses the mobile phones as a ubiquitous behavioral study platform to automatically recognize user activities.

First we propose a framework that tracks users in real time to capture temporal, spatial and customized attributes of their activities. We design the framework to be portable across various mobile devices while solving the crucial challenges of conserving power, network usage and privacy.

Next, using the framework we analyze user data collected over a span of several weeks. We use clustering techniques and error correction methods to train our system by identifying and classifying user activities. We facilitate the training process by providing location specific visual aids which allows customized user labeling of activities. Based on the knowledge gained in training we then discover repeated activities in the lifestyle of the user.

Finally, we propose methods to rapidly disseminate the acquired information by integrating with social networks and publishing it as map-based mashups. We also represent summarized information as a time-lined storyboard that serves as a visual window to look into the common activity patterns of the user.

To My Parents

Contents

List of Figures

Acknowledgements

My first words of gratitude go to my advisor Dr. Samir R. Das for his inspiration and guidance during my research and study at Stony Brook University. Thank you very much for fostering the roots of research in me and always providing unrelenting support. Dr. Das: It has been my pleasure and privilege to work with you and I treasure the insights I have gained along the way.

I am grateful to Dr. I.V. Ramakrishnan for being my committee chair. Fondly referred by students as Dr I.V., the encouragement he has given to students as the Graduate Program Director is immense and his advice highly motivational.

I will always cherish the moments I spent working on the Campus Bus Tracking project with Dr. Jennifer L. Wong. I am indebted to you for all your insightful comments and constructive suggestions during the project. Working on building the project really opened the doors for me to realize my passion of taking technology forward to people. Thank you Dr. Wong, for being my thesis committee member and helping in shaping my thesis.

I would also like to thank Dr. Jie Gao for her enthusiastic advice and encouragement during the Bus Tracking Project. Be it doubts with algorithms or needing some inspiration, Dr. Gao's door was always open.

I also extend my gratitude to my friends at the WINGS lab: Kanthi, Pralhad, Ritesh, Anand Prabhu, Rajendran and Utpal. Thank you for your support and encouragement during my thesis.

This thesis would not have been possible without the love and support of my parents. This thesis is dedicated to them!

Chapter 1

Introduction

Location based services have become extremely popular in the recent years in a variety of contexts. These services provide information that has been created, compiled, selected or filtering primarily based on the current locations of the users. The primary focus behind location based services is that they can be used to augment several other existing services like telephony (location based billing), delivering emergency alerts or advertising. Thus location based services today represent the most powerful means to personalize mobile services.

With the modern day mobile phones becoming more powerful with integrated GPS and other sensors like accelerometer, touch sensors etc, the mobile phone has become a solid platform to deliver location based services. Wireless carriers and their partners are developing a host of new products, services, and business models based on location based services. As a result of multidimensional advantages provided by location based services wireless carriers have started treating location based services as the 'third asset' beyond voice and data.

Though the use of location information in traditional and new services market has become ubiquitous there has been little development towards using the location information for personal use. On the personal front, characterizing user behavior by harnessing location information provides an interesting challenge. Such a study can lead to recognizing activities in the lifestyle of the user and share such information with others. This behavioral study has a great social significance from the perspective of a trusted group or social clique.

Our work strives to utilize the ubiquitous nature of mobile phones equipped with sensors by using them as a platform to study and predict activity patterns of the user. Making use of mobile phones represents a feasible approach towards activity recognition as most of the real world activities and interactions can be studied using the sensors in modern mobile phones. A comprehensive evaluation of the different sensors on commodity cell phones and the aspects of real world interaction that they cover are studied in [13].

Using the cell phones as a behavioral study platform we present a framework to automatically recognize user activities that can be shared through social networks.

This scheme has two major incentives, first is that such cell phones are ubiquitous and convenient and secondly there is very little manual effort in this scheme.

In our work first we focus on solving the issues concerning the design of real time framework to capture the necessary data for the analysis from the users. Issues like conservation of battery power, efficient message exchanges and scalability are crucial aspects of our design. We present our solution and the architecture that distributes the computing required between the client applications and a central server in Chapter 3.

We present our techniques and algorithms in segregating activities based on collected data in Chapter 4. We have used clustering techniques to classify activity locations and show that it is possible to train our system very effectively with a subset of data collected. The Chapter also explains how we interact with the user for identifying activities.

Chapter 5 explores our experiments with collecting data from the user and the analysis done with the data. The chapter details our mechanisms to identify erroneous data and resolving ambiguity in identifying activities. We also show how error correction is done based on user feedback.

Chapter 6 focuses on injecting the learnt patterns into social networks for automatic and controlled dissemination of the knowledge gained by our tool. Also we deal with summarizing the information learnt and how it can be visually represented as a story board.

The conclusion of our study and future work is presented in Chapter 7.

Chapter 2

Problem Formulation and Related Work

2.1 Goals of Activity Recognition

Activity recognition aims to recognize the actions and goals of one or more agents from a series of observations on the agent's actions and the environmental conditions. Activity recognition has been predominantly treated as a classification problem [19]. Recognizing user activities help in building context awareness of applications that help to provide seamless and transparent services to users. Also recognizing activities would be pivotal in modeling user behavior in any ubiquitous computing system.

Such recognition systems could have profound conceptual and practical implications in terms of applications. Examples of such applications include providing care for the elderly or physically challenged through automatic supervision.

2.2 Relevant Work

Several works [8], [14] have explored recording of significant events in user's lives and building systems that recall such events.

The predominant aim behind these works has been to aid human memory by recalling events with the help of computers. They can record e-mail and chat conversations, documents, location information, photographic, audio, and video audio content using cameras and microphones, and many other types of personal and environmental data capture.

Choudhary et Al [3] have studied activity recognition by using small wearable device designed for embedded activity with the aim of broadly supporting context-aware ubiquitous computing applications.

Other sensors like accelerometers have been studied in predicting the physical activity of the user. Randell and Muller [17] used a single biaxial accelerometer for physical activity recognition. They classified primarily into six physical activities (walking, running, sitting, walking upstairs, walking downstairs, and standing) utilizing a neural network.

Most of the studies described above used specialized and wired hardware sensors used to predict activities. Such hardware is often uncomfortable to wear and does not provide incentives for users to follow this approach in real life.

Morikawa et Al [16] used cell phones equipped with RFID transmitters/receivers to build profile information of users. The events are linked through semantic web analysis and events can be recalled using a remembrance viewer application. Finally Miluzzo et Al[15] describe a way of injecting the social presence of users to a variety of systems.

2.3 Our Approach

We intend to build a framework that does activity recognition using only sensors in commodity mobile phones. We discount the need to carry specialized hardware or wearable sensors and harness the power of ubiquitous presence of mobile phones instead. We also build our solution to be portable across different breeds of mobile phones and can work straight out of the box after installation. While we extensively make use of integrated GPS as our sensor to detect contextual activities, our framework will have the provisions to be easily augmented to work with other phone sensors like accelerometer or even Bluetooth and Wi-Fi radios to perform fine grained and indoor activity recognition.

Our approach also differs in the way that it not only captures both temporal and spatial attributes automatically but also captures customized attributes like status messages or description tags by directly interacting with the user. These custom attributes can serve dual purposes, to both segregate information having same temporal and spatial characteristics for a single user to provide fine granular information and to also aggregate such characteristics having the same custom attribute among groups of users.

We solve the activity recognition problem by requiring minimal training data by identifying the activities belonging to locations by using a clustering approach. The learnt spatial contextual information of one user's activity can be directly applied to other users training and hence our method would directly support cross training among trusted group of users sharing activities taking into account both privacy and trust constraints.

Rather than focusing on specialized environments with constrained data collection we train our framework with a more generalized drive trace collected across several weeks and show that activity recognition can be directly performed on such traces. Thus we provide a method to plug and play the source of training data in our solution and thus we can make use of alternate data collected from other source devices and use this to analyze activities from newer data collected by mobile phones for the same user.

We also seek to support an elegant way to rectify any erroneous data samples that we encounter due to fragilities in the automatic data collection process. We automatically identify such results and provide a mechanism to allow the end users to correct the results using a simple drag and drop method. Such a process is facilitated by making use of displaying the error range for the location, street view and reverse geocoded address.

We present schemes to disseminate the learnt activities through social networks which help to automatically harness the power of collaboration among a group of trusted users. This helps to seamlessly integrate common activities among a trusted set of users and the common activities are identified and summarized. We can easily weave the real world social networking paradigms into activity recognition directly as a result.

Finally as a result of activity recognition we build rich user specific summarizations to deliver the results across a span of several days that reflect both temporal statistics like time spent on the activities and also capture interactions with other users across activities. Summarizations can range from analyzing a single day's activities to learning about behavioral patterns across a number of days.

2.4 Using Location for Activity Recognition

We use the user's location as the primary driver for activity recognition in our solution. Identifying the location of an event helps to establish possible activities of the user at that particular location. For example a location like office tells that the activity of the user is @Work with high probability. Moreover over multiple days of the year a users behavior will primarily tend to repeat based on a few selected locations where typically the same set of activities are performed again and again. Thus obtaining the location is a critical requirement for activity recognition in our solution. We deal with techniques and methods to obtain relevant location in this section

2.4.1 Positioning Systems

A positioning system helps to identify the spatial position of a target. Any positioning system achieves its functionality through measurement of one or several *observables*, for example, angles, ranges, range differences, or velocity. Such an observable usually reflects the spatial relation of a target relative to a single or a number of *fixed points* in the surrounding environment, where a fixed point denotes a point of well-known coordinates. A positioning method delivers a target's position with regard to a descriptive or spatial reference system.

Examples of positioning systems include GPS, Cell Tower GSM or Wi-Fi based systems.

2.4.2 Characteristics of Positioning Systems

Accuracy: The position returned by the positioning system must be as close to the true position of the target.
Availability: The system should be available at different possible locations and scenarios like rural, urban etc
Resource Usage: The positioning system typically makes use of processing power at the node consuming electricity
Latency: There is a finite amount of delay that the positioning system encounters between the request for position and obtaining the position.

2.4.3 Global Positioning System (GPS)

The most prominent positioning infrastructure is the GPS which is a standalone infrastructure based on satellites. The GPS uses a constellation of between 24 and 32 Medium Earth Orbit satellites that transmit precise microwave signals that enable GPS receivers to determine their location, speed, direction, and time.

A GPS receiver calculates its position by carefully timing the signals sent by the constellation of GPS satellites high above the Earth. Each satellite continually transmits messages containing the time the message was sent, a precise orbit for the satellite sending the message (the ephemeris), and the general system health and rough orbits of all GPS satellites (the almanac). These signals travel at the speed of light through outer space, and slightly slower through the atmosphere. The receiver

uses the arrival time of each message to measure the distance to each satellite thereby establishing that the GPS receiver is approximately on the surfaces of spheres centered at each satellite. The GPS receiver also uses, when appropriate, the knowledge that the GPS receiver is on (if vehicle altitude is known) or near the surface of a sphere centered at the earth center. This information is then used to estimate the position of the GPS receiver as the intersection of sphere surfaces. The resulting coordinates are converted to a more convenient form for the user such as latitude and longitude for being displayed.

Advantages of GPS

The advantages of the GPS system are the global availability and comparatively high accuracy of position fixes. GPS receivers are present in most modern day mobile phones which makes this system the best choice for obtaining the location of the users in our solution.

Disadvantages of GPS

However since GPS relies on satellites a common disadvantage is that due to shadowing effects positioning needs a line of sight which is not possible indoor. The other major challenge with making use of GPS is that the GPS receiver consumes high power which is not a major concern with GPS devices in an automobile where it is powered by the automobile circuitry. However for our scope the battery power of the mobile phone is limited and hence we propose solutions to conserve power while making use of GPS.

Chapter 3

System Architecture

We seek to build a highly interactive and end-end system that does automatic activity recognition of users activities. Our systems architecture which is highly flexible and scalable solution to activity recognition is explored in this chapter.

3.1 Research Goals

The major goals based on which the system is designed are discussed in this section.

3.1.1 Use COTS Hardware

The key element that distinguishes our work from other works in the area is to directly make use of commodity off the shelf mobile devices. We do not restrict our work to specialized hardware or particular platform requirements.

3.1.2 Collect Required Data Automatically

Our system is aimed at collecting the required data without any intervention from the user by operating in the background. We do not require any manual inputs from the user to continue recognizing the activities.

3.1.3 Support Customized Attributes

Beyond capturing temporal and special attributes we aim to support customized attributes for activities. This results in fine grained activity recognition. A classical example would be visiting a restaurant which could be either for ‘dine in’ or ‘take out’ food both of which would fall under the larger category of ‘dinner’ but with a suitable annotation can mean two different activities.

3.1.4 Efficient Resource Usage

Our framework seeks to minimize resource usage both in terms of power and network usage as we are using mobile phones to collect data. This is a critical aspect to the framework because we would like to do precise prediction of activities close to the ground truth by utilizing as much fewer resources as possible.

3.1.5 Spread Learnt Information Automatically

We design the system so that it automatically publishes the learnt information automatically by integrating with social networks. This promotes implicit communication between groups of users. Another goal of the system is to automatically take care of privacy and trust issues among other users of the system by allowing fine grained control on what information has to be shared and what is not to be shared.

3.1.6 Summarize the Story

We would like to capture the activities recognized from the collected data in the form of a storyboard. Our approach is to present this information along a graphically rich timeline that can be used as a window to recall any interesting aspects of activity recognition. Moreover we strive to automatically classify group of activities and represent them as closely as possible to a natural language description like “a busy day at the office” etc in the future.

3.2 Hardware and Software Information

3.2.1 The Client Mobile Phones

In our work we made use of basically two types of mobile phones that differ significantly in the platform used. We present the hardware details below

HP iPAQ hw6900 Mobile Messenger

The iPAQ 6900 is a Windows Mobile 5.0 based pocket phone. The phone is powered by Intel PXA270 Processor running at 416 MHz). The iPAQ is equipped with GPRS/EDGE technologies provide broad high-speed connectivity over the mobile phone network to access data. Integrated Wi-Fi (802.11b) enables high-speed data rates through access points. It has an integrated GPS receiver that can be used for location sensing.

Figure 3.1: HP iPAQ Smartphone 6900.

Openmoko Neo Free Runner 1973

The Neo FreeRunner is a Linux-based touch screen smart phone ultimately aimed at general consumer use as well as Linux desktop users and software developers. It is based on open source initiative to develop an open phone so that Linux users and software developers will appreciate the total freedom they have to use and design software for the FreeRunner.

The Neo FreeRunner is the second phone designed to run Openmoko software. It is powered by a 400Mhz Samsung ARM processor. It contains the following sensors and radios

- Built-in 802.11b/g Radio (Atheros chipset AR6001 Flash version)
- 2 built-in Tri-Axis sensors (ST accelerometer LIS302DL)
- Built-in GPS Radio with internal antenna having a -130 dBm signal strength for tracking
- Tri-band GSM and GPRS

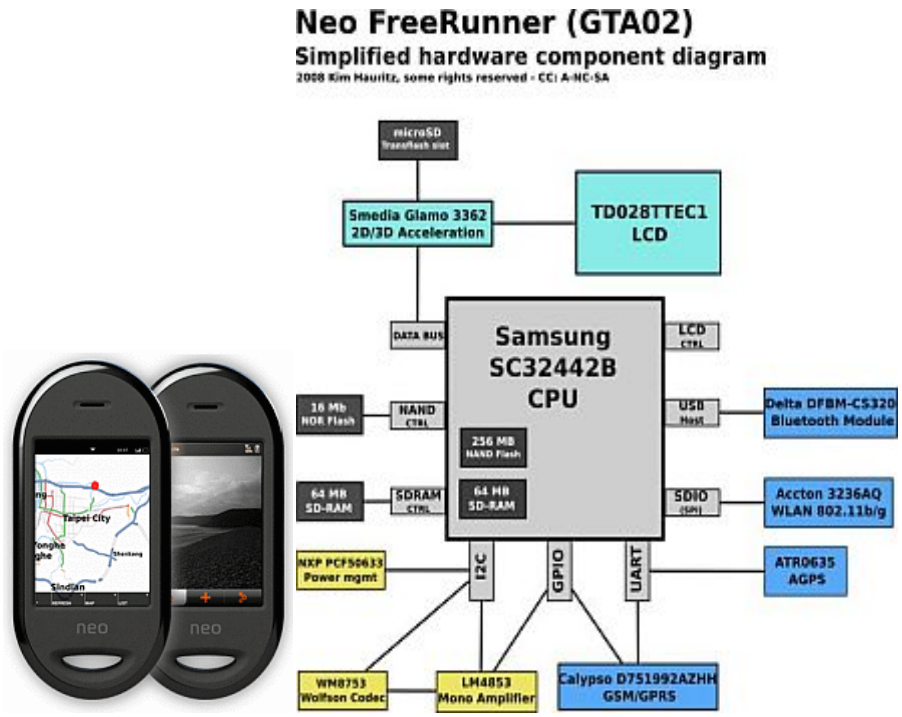

Figure 3.2: Openmoko Neo FreeRunner phone. (a) The phone running Openmoko Linux, (b) Component diagram showing integrated sensors

3.3 Design

Our design is driven by a client server architecture, where all mobile phones act as client devices run a client application that pushes all location updates to the server. The server is the central point that collects all location updates from clients. The design of the entire framework is shown in Figure 3.3

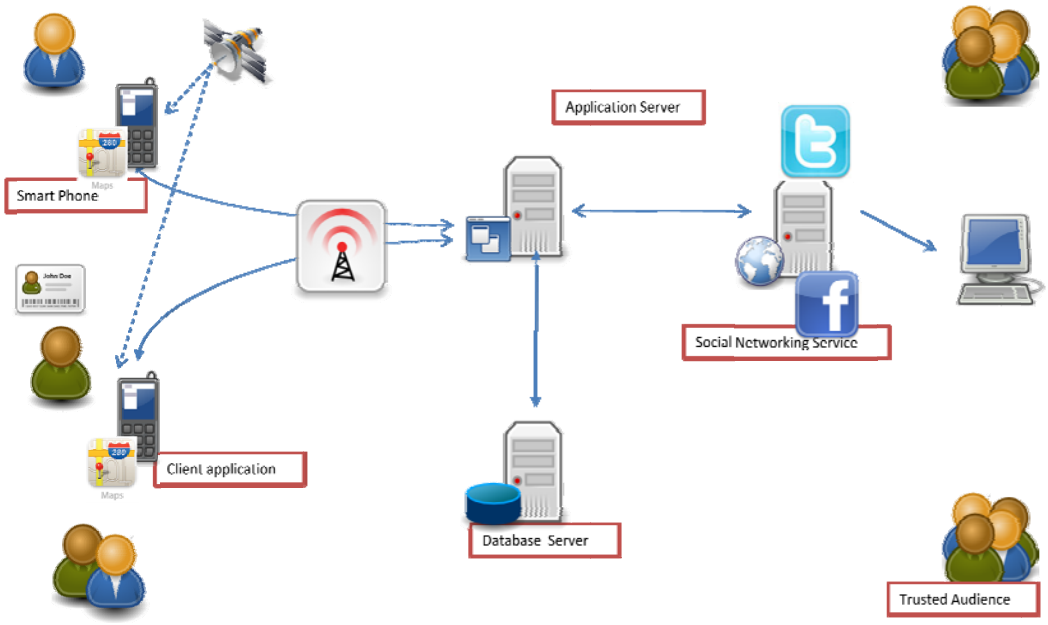

Figure 3.3: Framework design diagram showing the components.

3.3.1 Components of the System

Client Application

The client application runs on the mobile phones and fetches location updates from the user. Each location update consists of user-id, latitude, longitude, timestamp, status messages and annotation information. The client application also displays a map interface which shows real time tracking of users in the locality.

Application Server

The application server is the centralized point in our design. It is responsible for talking to all clients and collecting information. It stores the information persistently to a database.

The application server also talks to social networking services to establish trust and privacy relations to user. It interacts and publishes learnt information on social networks. It also delivers information on other users to the clients. The application server runs apache web server (v 2.2.9) on Ubuntu 8.10 Linux (2.6.27-9-generic kernel).

Database Server

The database server acts as persistent storage for all data involved in the system. It also helps in transforming physical activity relations to logical database queries in the system. E.g. It helps in figuring out event like most recent activity where user1 and user2 where present together say a meeting. The database server we use is MySQL version 5.0.67

3.4 Developing the Client

The client is responsible for collecting all the data required for analysis, interacting with the user to display real-time maps and gather inputs like status messages and labels. The client runs on the mobile phone of the user.

3.4.1 Client Design Research Challenges

Portability

The major challenge we faced is the choice of a programming platform that would enable the client application developed to be deployed readily on several commodity cell phones. Though mobile phones are very diversified according to various segments of markets, the recent trend has been that more mobile phones are becoming smart in nature with integrated sensors like GPS, accelerometer etc. These phones are equipped with mobile processors that have the adequate processing power to very easily multitask applications and are well suited for our data collection aspects.

The common programming platforms available are heavily dependent on the type of Operating system that runs on the phones. Some major platforms include Windows CE, Symbian, Linux variants and recently Android and iPhone.

Our Approach

In our work we implement our client side application that runs on mobile phones on the Java 2 Micro Edition Platform (J2ME). Java 2 Micro Edition (J2ME) is Sun's

version of Java aimed at machines with limited hardware resources such as PDAs, cell phones, and other consumer electronic and embedded devices [20]. J2ME is aimed at machines with as little as 128KB of RAM and with processors a lot less powerful than those used on typical desktop and server machines.

J2ME Profiles and Configurations

J2ME actually consists of a set of profiles. Each profile is defined for a particular type of device like cell phones, PDAs or microwave ovens and consists of a minimum set of class libraries required for the particular type of device and a specification of a Java virtual machine required to support the device. A profile is just the specification which is implemented through a configuration.

In our work we make use of the 'The Mobile Information Device Profile (MIDP)' which is a profile for mobile information devices, such as cellular phones and two-way pagers, and PDAs. The configuration that is used is Connected Limited Device Configuration (CLDC) which is an implementation of MIDP for small, resource-constrained devices such as Palm OS devices. When coupled with MIDP profile, the Connected Limited Device Configuration gives developers a solid Java platform for creating applications for consumer and embedded devices.

J2ME Benefits

- J2ME is widely supported across various platforms
- J2ME can be easily emulated on a PC during the development stages and the built software can be easily uploaded and installed on the phones
- Because of the rich set of emulation features available, application development does not need physical hardware or mobile phones to test them in the initial stages of development.

Selecting the Java Virtual Machine

There are various different JVMs supplied by vendors for several different platforms. In our solution we make use of the IBM j9 Java virtual machine [9]. The design of the J9 VM has been aimed at portability to different platforms, as well as scaling from mobile phones all the way to zSeries mainframes.

Our solution is directly portable and can be run on other JVMs that support J2ME specifications.

3.5 Client Application Implementation

As described above, the client application is responsible for interaction with end users and by collecting location updates from the GPS on the mobile phone.

In our approach, the client application is written in J2ME for reasons outlined in the previous section. In this section we look into details of our implementation of the client application.

3.5.1 J2ME MIDlet Suite

The client application is written as a MIDlet suite. A MIDlet suite is a Java application framework for the Mobile Information Device Profile (MIDP) that is typically implemented on a Java-enabled cell phone or other embedded device or emulator.

The MIDlet suite consist of two major components

- A Java Application Descriptor (JAD) which describes the MIDlet suite. The description includes the name of the MIDlet suite, the location and size of the JAR file, and the configuration and profile requirements. In our requirements we specify access to the GPS and the Internet as a profile requirement. This triggers a pop up message for the user to allow connection to the Internet when trying to reach our server.
- A Java Archive (JAR) file which contains the actual MIDlet code that will be executed

The MIDlet has a state and our programming is event driven programming model. The method inherited from the MIDlet base class (provided by J2ME) allows us to create, start, pause, and destroy the MIDlet. Our MIDlet implements these methods to update its internal activities and resource usage based on the various events we encounter when our application runs. Our MIDlet makes use of the LCDUI to draw the GUI components on the mobile devices screen.

3.5.2 Multithreading

Our client application has three major functionalities that can be very easily represented as concurrent threads running inside the MIDlet. The threads represent the functionalities below

- Poll the GPS periodically to read the current location of the user
- Push any location update to the server
- Retrieve the locations of other users from the server
- Draw the GUI and the map interface on the screen

These functions represent the unique flows of execution within our application. Out of the functions, functions 1 to 3 are I/O Intensive and the GUI is CPU intensive. This gives us a good mix to create threads and schedule them intelligently so that performance is not affected. Also functions 2 and 3 talk to the server over network so have to take care of asynchronous error condition that can arise out of the network. We multiplex these functionalities onto a single thread abstraction so that we can both piggy back network operations and handle error conditions through a common channel. As a result we have the following threads designed to take care of the four functionalities described above

- Poll the GPS - “location_Thread”
- Push any location update to the server – “network_Thread”
- Retrieve the locations of other users from the server – “network_Thread”
- Draw the GUI and the map interface on the screen – “ui_Thread”
- Handling signals and commands from the mobile phone – The base midlet or the “master thread”.

3.5.3 Client GUI

Our GUI is implemented through the MIDlet LCDUI API. The GUI has a simple screen based approach where a single Displayable is always active at a time in the application user interface. Java LCDUI API provides a small set of displayable objects common in mobile device user interfaces: List, Alert, Textbox, Form and Canvas. Canvas is a low-level graphics surface for which an application has full control over what is rendered to it. We make use of the Canvas to draw our UI map object on the screen.

LCDUI also helps us to abstract operations called Commands. Common types are BACK, EXIT, ITEM and SCREEN. The idea of the command abstraction is to make applications more portable between various different mobile devices. We make use of the commands provided by LCDUI to interact with the user in our GUI.

The GUI initialization asks the user for the name on startup. This helps to uniquely identify the user within our framework and deal with the data from the user in the appropriate way. Once we obtain the user name and identify the user, the user is greeted with a map showing the current location of the user. The application is then completely menu driven based on the nature of interactions and input choices selected by the user.

3.5.4 Displaying the Map

The Map is a critical component of the GUI on which the various elements of our application are shown. The map is the base object on which we show other information relevant to the user. Some of the information shown on beside the map is

- The Current location of the user
- Locations of users nearby
- Ability to home in or track any user who has authorized such tracking
- Status messages of each user etc
- This information is displayed similar to Google Maps with the help of Markers and Information windows. We term all this information as overlays displayed on top of the base map.

We draw the base map on the canvas and set it as the current item that is to be displayed on the mobile phone screen. We make use of the J2ME Maps API to achieve common functionality needed for our application [1]. The J2ME Map is a free interface for non commercial use that helps us to perform common activities like the following.

- Display a Google Map on the screen
- Change the views of the map between normal and satellite
- Zoom In/ Zoom out functionality

Apart from the basic functionality we also have added functionality like switching between Google, MSN and Yahoo Maps. Most importantly is the ability to extend the API with our own data. Figure 3.4 shows the screen shot of our client Map application running on the emulator on a PC.

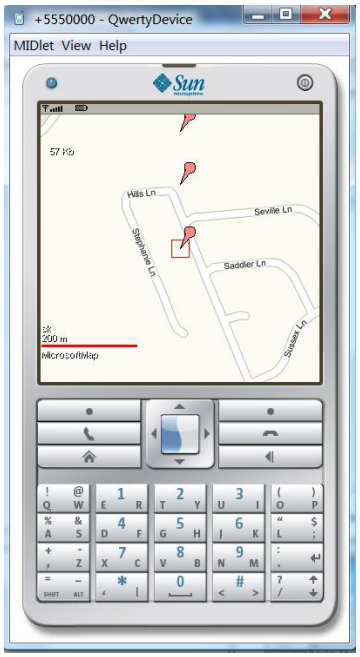

Figure 3.4: Client Application.

3.6 Obtaining Location Updates

With the suitable platform identified to build our client we now turn our attention to solve the problem of propagating the location information from the clients to servers.

3.6.1 Challenges

We first look at the challenges associated with obtaining location information from user's mobile phones. The major challenges are follows:

Real time Changes

Location updates happen in real time and hence this information should be obtained as quickly as possible from the user. However the mobile phone being an unreliable data source, obtaining the real time information is not always possible and hence it is not a critical aspect of our system design. This makes the location updates to be soft real time in nature inside our framework and the framework is designed to tolerate any failures in obtaining this information.

Discontinuous

People tend to be engaged in an activity for at least a non trivial duration of time which means that the location of the user remains static during the activity. The system should not try to poll often for obtaining each user location as this would

have adverse effects both in terms of battery life of the mobile phones and the cost associated with each message transmission. We solve this problem by building triggers into the system that help us to monitor for updates without any adverse effects.

Enormity of Data

While the client does not store any data collected over a period of time, the server tends to aggregate all data collected on behalf of several users for analysis. It is hence very important for the client to minimize the amount of data being pushed to the server.

3. 7 Design of Client and Server Application Interaction

As mentioned already the client and server interact through http protocol to make sure that there are no lost updates. The client consumes a web service exposed by the Application server.

3.7.1 HTTP Connection Semantics

Typically mobile data usage is not free and is charged by the all carriers according to usage. Most telephone carriers in fact employ soft caps on data usage by applying QoS policies on mobile data bandwidth. Carriers have finite resources and typically they balance between voice and data traffic. As a result the cost of accessing data based services is charged to the user using the application.

Since the most common and widely prevalent cost model is to charge the end user for every byte of data used this becomes a key factor in designing our client server interaction semantics. HTTP typically operates in two different modes based on the version of the protocol implemented [18].

Persistent Connection mode

By default HTTP version 1.1 uses persistent connection mode where a connection established to server is kept open until it is explicitly torn down. This avoids the need to do TCP handshakes every time to establish a HTTP connection with the same server.

Multiple Connection modes

In HTTP version 1.0 the connection is closed after a single request/response pair. This means that a TCP handshake should occur every time we need to establish a connection with the server.

The following figure illustrates the difference between two modes.

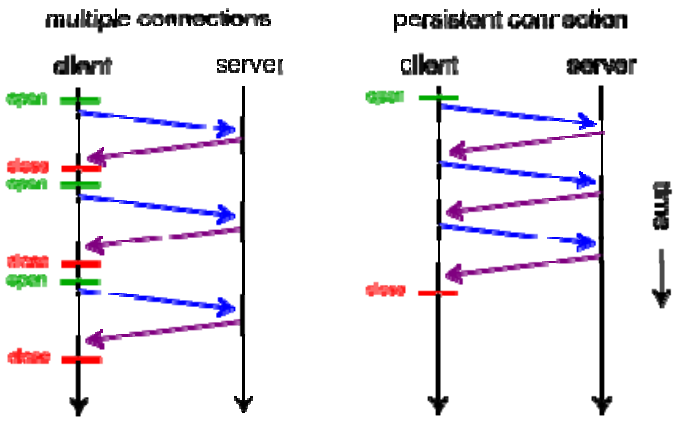

Figure 3.5: HTTP Connection Modes. (a) Multiple Connections Mode,

(b) Persistent Connection Mode.

3.7.2 Our Approach

In our solution, we do on demand HTTP connections because of the following factors

- A persistent connection forces the network adapter to remain on when in use causing a significant drain in battery power.
- Because the connections to mobile phones are intermittent in nature it is unreliable and hence making use of a persistent connection does not offer significant benefits.
- Persistent connections require non zero CPU and Memory usage. Though the factors are tiny they are important on mobile devices.

3.8 Optimizing Network Usage

To reduce the amount of data transmitted we add various optimizations to our client and server. This makes sure that we conserve network resource usage on our client application.

3.8.1 Client Side Caching

We display the map on the client application to display users and other relevant information on the GUI. The map is made up of tiles of images that are stitched together to present the illusion of a larger image. This helps in scrolling the map in any direction where only the particular new tile is redrawn to give a seamless appearance of a continuous map.

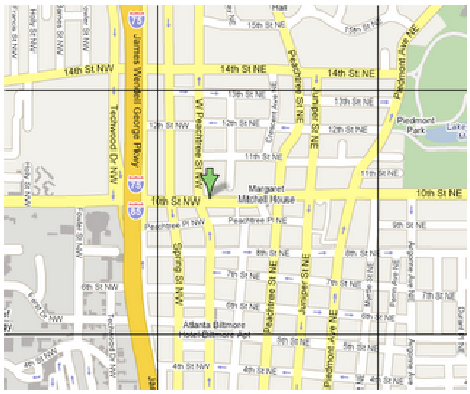

Figure 3.6: Rendering the map on client GUI as a set of tiles

Google Maps uses an x, y coordinate system combined with a zoom value to specify the tiles to retrieve from the server [7]. These coordinates are calculated using a reference central point which is the "center" of the Google Map. The current center point is a point in Kansas its Lat/Long coordinate are (39.5N, 98.35W).

Each x and y value represents the top left corner of a tile. This is passed on the url request with the zoom level to the web server. An example request for fetching a tile looks like http://mt1.google.com/mt?n=404&x=621&y=760&zoom=6.

The corresponding tile is shown below in the figure

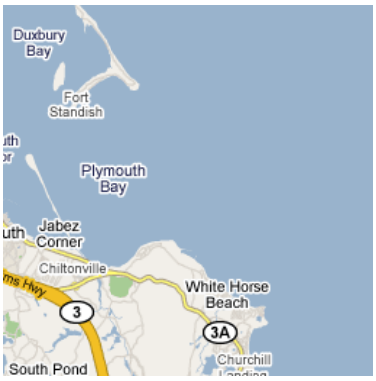

Figure 3.7: A single map tile for the values x=621, y=760 and at zoom level 6.

Each of these images is static content and is approximately 30 kb in size. We cache these images on the mobile device to avoid requesting the same tile again from the server. The images are stored in the cache as {x-y-n-z} format so that they can be easily identified. When the zoom level is increased each tile gives 2x2 tiles at the next zoom level

This helps in avoiding redundant network transfer as typically user activities are within some locality of the map. This also makes the GUI extremely responsive and quick by rapidly loading the tiles.

3.8.2 Optimizing Transport using Compression

Each location update sent to the server is sent as an XML file containing the following tags: user-id, latitude, longitude and timestamp. XML tags are similar to html tags but having a strict validation rule that a <tag> should be ended with a corresponding </tag>

We minimize the redundant data bytes transferred by compressing the xml data exchanged in our messages. However any compression algorithm consumes CPU overhead for the compression process. Since we deal with mobile devices where CPU power is a scarce resource we make use of light weight compression schemes. In this regard we make use of Jzlib a light weight zlib port for Java devices [10]

3.8.3 Piggy Backing Transfers

A new location update has to be posted to the server web service by the client application. However the location update will usually trigger the map GUI to be redrawn to the new location. This results in obtaining a new tile from the server for the new location.

We optimize the number of connections by multiplexing the request for a new tile image and the location update to a single HTTP request. This is done by piggybacking the location update xml data on to the GET request for the map tile.

Example of Piggybacking is as follows

Request: HTTP GET http://www.wings.cs.sunysb.edu/tiles(x,y,Zoom)

Data:

<location>
<latitude>

40.8988
</latitude>
<longitude>
-73.4129
<userid>…
</location>

This helps to avoid establishing concurrent and new on demand http connections to send location updates. The image tile is only requested if the tile is not there in the cache. If the image is in the cache a HEAD HTTP request is issued with the E-Tag of the cached tile to make sure the content in the cache is still valid.

3.9 Optimizing Power Usage

We observe that making use of GPS to obtain location information is a significant factor in consuming power on the mobile phone. While we cannot discount the use of GPS to obtain location information, we would want to conserve as much power as possible on the phone.

A polling based solution tends to keep the GPS on at all times and thus it drains all the power on the mobile phone. To avoid this situation we turn on the GPS device only when we find the necessity to gather a location update. At all other times we put the GPS in power off state. This is done by closing the device on the mobile phone which triggers the appropriate power control API to turn the GPS off.

GPS Cold Start

The issue with keeping the GPS off when not necessary is a direct hit in performance. This is because each time the GPS is turned on the receiver has to lock on to the satellites to obtain the position information. This causes a cold start that usually takes around 50 – 60 seconds to get a position fix. This makes it infeasible for our needs where we would want to get the position fix as soon as possible and send it to the server.

In order to overcome this issue we persist the last known position fix and inject it into the GPS device driver when turning the GPS on again. This helps in obtaining a rapid position fix (< 3 seconds) which is then updated to the server.

Chapter 4

Recognizing Activities

User activities are contextually associated with locations or places. In a typical day any user can be expected to perform all activities across set of known locations like ‘home’, ‘office’ etc. It is important to identify the location of any activity that is occurring in the lifestyle of the user to reasonably predict the activity going on. So our major focus is to automatically identify these locations from the collected data so that the user’s activities can be recognized.

4. 1 Our Approach

The next problem we face is to automatically identify locations for various activities from the continuous stream of collected data. In order to achieve this we have to first classify the locations automatically so that the user can easily identify and label them.

Our approach to identify distinct locations encountered in the collected data is to use clustering techniques to segregate the activity points to distinct clusters.

4.2 Representing Locations in Cartesian plane

We transform the latitude and longitude to a Cartesian plane so that it is easier for us to apply conventional clustering algorithms and calculate distances easily.

4.2.1 A Location in Geographical System

When we represent the location of objects in three dimensions there are several different types of coordinate systems that can be used to represent the location with respect to some point of origin. A location that is collected by the GPS is typically a latitude and longitude that represents a point in the Geographical Coordinate System which is a spherical coordinate system.

The spherical coordinate system uses three parameters to represent a point in space, a radial distance (*r*) from a point of origin to the point in space, a zenith angle (θ) from the positive z-axis, and azimuth angle (Φ) from the positive x-axis. In Geographical Coordinate System the radial distance is the radius of the Earth, the zenith angle is the latitude, and the azimuth angle is the longitude. The origin is considered to be the center of the Earth. The Cartesian coordinate system is the standard x, y, and z coordinate system that is commonly used to represent a point in space. The following figure gives a graphical representation of the spherical coordinate system. It also points out the Cartesian coordinate system equivalent dimensions.

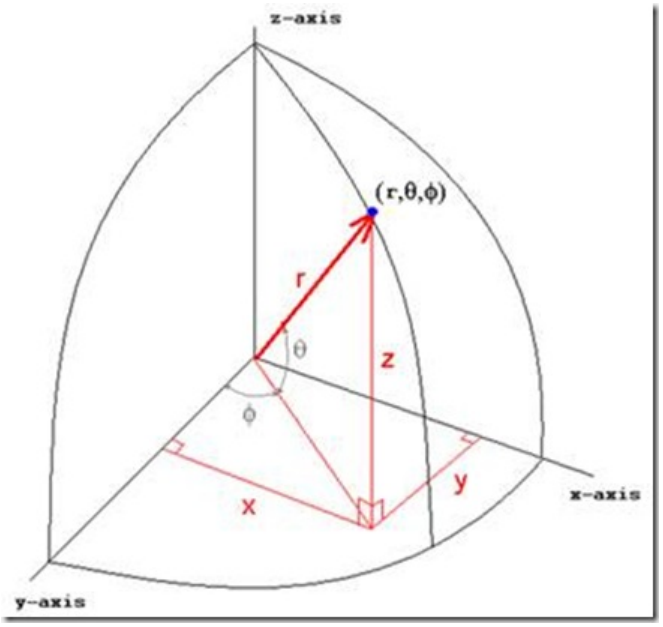

Figure 4.1: Spherical and Cartesian Coordinate System.

4.2.2 Conversion Semantics

We transform the Spherical coordinates to x, y, z components of the Cartesian coordinate system. The following equations give the transformation between the spherical and Cartesian coordinate system

$$z = r \sin\theta$$

$$x = r \cos\theta \cos\emptyset$$

$$y = r \cos\theta \sin\emptyset$$

We also need to convert back i.e. do the reverse to display clustered points in the Spherical system. Using Pythagorean Theorem we calculate the length of r as

$$r = \sqrt{x^2 + y^2 + z^2}$$

In order to convert back we first determine the angle θ as

$$\theta = \sin^{-1}(\frac{z}{r})$$

and the Φ angle as

$$\phi = \tan^{-1}(\frac{x}{y})$$

Using the above relations we can easily transform between the Spherical and Cartesian systems and viz.

4.3 Clustering

We can define clustering as "the process of organizing objects into groups whose members are similar in some way".

A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. So, the goal of clustering is to determine the intrinsic grouping in a set of unlabeled data.

4.3.1 Goals of Clustering

The major goals of our clustering would be to

- Scalability
- Minimal requirements of domain specific parameters
- Interpretability and usability
- High dimensionality support is not much of a concern to our clustering approach as we only deal with a co-ordinate based system with three axes to represent points.
- However we would need the algorithm to converge relatively quickly into a solution to identify the unique points.

4.3.2 Using K-Means to Identify Locations

4.3.2.1 K-Means Clustering Algorithm

K-means is one of the simplest unsupervised learning algorithms that solves the well known clustering problem. K means clustering algorithm was developed by J. Macqueen (1967) and then by J. A. Hartigan and M. A. Wong around 1975.

K-means clustering is an algorithm to classify or to group our objects based on attributes/features into K number of group. K is positive integer number which is the number of clusters and is a known priori. Grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid of the K groups. Thus the purpose of K-mean clustering is to classify the data.

The main idea is to define k centroids, one for each cluster. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending we need to re-calculate k new centroids as new centroids of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has thus been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

We call this “unsupervised learning” because the algorithm classifies the object automatically only based on the criteria that we give (i.e. minimum distance to the centroid). We don't need to supervise the program by saying that the classification was correct or wrong.

To summarize these in the form of steps

1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

4.3.2.2 Objective Function

We use a squared error function as the objective function which we try to minimize when we run K-Means clustering.

The objective function we use is given by

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} (x_i^j - c_j)^2$$

where $(x_i^j - c_j)^2$ is a chosen distance measure between a data point x_i^j and the cluster centrec_j, is an indicator of the distance of the *n* data points from their respective cluster centers.

4.3.2.3 Implementation Choices

K-Means does not specify the way to initialize the mean. We adhere to the most popular way of choosing k random points from the input samples.

The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. To avoid this we try a number of different starting points.

It can so happen that the set of samples closest to K_i is empty, so that K_i cannot be updated. We handle this appropriately as a special case.

4.3.2.4 Weakness of K-Means

While K-Means is effective in classifying the locations into distinct clusters it is not a general case algorithm that is very effective in practice. In particular it has the following weakness:

- The number of cluster, K, must be determined beforehand which is not known in our case
- K-Means is very sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the local optimum.
- We never know if there is a possible better solution for the same input data.

4.3.3 G Means: Normalizing K Means for K Normalized Cuts

The K-Means Algorithms tends to seek local rather than the global minimum solutions, and can hence get stuck at poor solutions. Also we would likely to automatically identify the value of k by learning based on input data and hence remove the constraint of specifying k as a parameter to K means.

Intuitively we would want our error function J within one cluster to follow a uni-modal distribution. To solve this problem we use the idea from G-Means algorithm

to learn the number k. For the ideal value of k the distance errors would follow a Gaussian distribution.

We start with k=1 and iteratively keep incrementing k until we obtain the lowest value of k such that all error distances follow the Gaussian distribution. To check for normality we use the Anderson-Darling test which can detect deviations from normality for the given data set.

4.3.4 Testing for Normality

The Anderson-Darling test is one of the most powerful statistics for detecting most departures from normality. The Anderson-Darling test assesses whether a sample comes from a specified distribution which in our case is the normal distribution. The formula for the test statistic *A* to assess if data $\{Y_1 < \cdots < Y_N\}$comes from a distribution with cumulative distribution function (CDF) *F* is

$$A^2 = -N - S$$

where

$$S = \sum_{k=1}^{N} \frac{2k-1}{N} \left[\ln F(Y_k) + \ln\left(1 - F(Y_{N+1-k})\right)\right].$$

The test statistic can then be compared against the critical values of the normal distribution to determine the P-value or Critical Value.

The Anderson-Darling test for normality is a distance or empirical distribution function (EDF) test. It is based upon the concept that when given a hypothesized underlying distribution, the data can be transformed to a uniform distribution. The transformed sample data can be then tested for uniformity with a distance test .

4.3.4.1 Procedure

We test for normal distribution of the variable *X* which is the error distance metric. Error distance is the distance of a point P_i belonging to Cluster K_i and is the distance between P_i and Centroid C_i of Cluster K_i.

1) We sort all X_i of a cluster C_i from low to high.

2) The mean, $\bar{X}$, and standard deviation, *s*, are calculated from the sample of *X*.

3) The values of X are standardized as follows:

$$Y_i = \frac{X_i - \bar{X}}{s}$$

4) With the standard normal CDF Φ, A^2 is calculated using:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)(\ln\Phi(Y_i) + \ln(1-\Phi(Y_{n+1-i}))).$$

5) A^{2*}, an approximate adjustment for sample size, is calculated using:

$$A^{2*} = A^2\left(1 + \frac{0.75}{n} + \frac{2.25}{n^2}\right)$$

6) If A^{2*} exceeds 0.752 then the hypothesis of normality is rejected for a 5% level test.

The following exception can occur in normal Anderson Darling Test calculation:

If s = 0 or any P_i = (0 or 1) then A^2 cannot be calculated and is undefined. However we never have s=0 as our k-means ensures that each cluster has at least one point associated to it.

4.4 Labeling the learnt locations

The next major activity in the training phase is to allow the user to give meaningful and contextual labels to identified locations. In order to do this we present to the user both the locations on a Map and also provide the user with additional information that can be used to easily and naturally identify the location.

When the user identifies the activity and the location the user can click the corresponding marker to select it. Clicking the marker shows information on the number of times the activity was performed and the average time spent on the activity. Once the marker is selected the user can then tag the location by entering a tag in the text area provided and by clicking the tag button. Tagged locations are identified by a separate color so that they can be easily identified.

Figures 4.2 and 4.3 illustrate the labeling process. It can be noted once the location is tagged the tag is shown in the informational window and the color of the marker is changed to red.

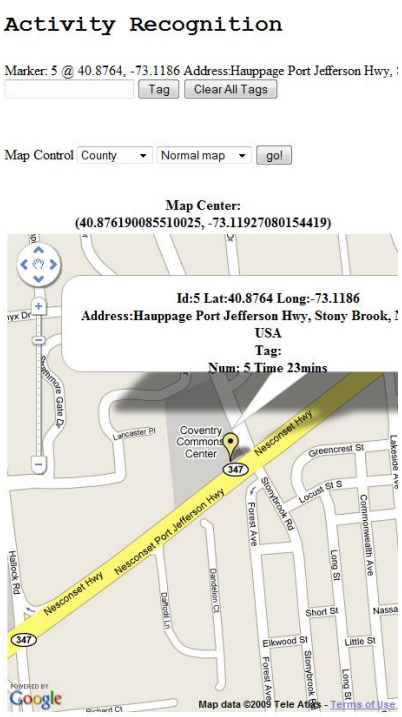

Figure 4.2: Untagged Marker selected for labeling.

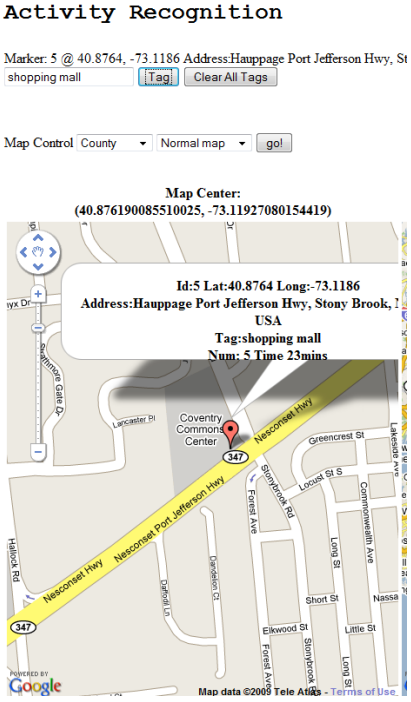

Figure 4.3: Labeling the activity by tagging.

4.5 Aiding the Training Process

4.5.1 Reverse Geocoding

A latitude, longitude in the form of 40.89,-73.10 is not very intuitive in practical terms to identify a location. In day to day life, we tend to identify the location primarily by an address in the locality where it belongs. An address like Computer Science Building, Lake Dr, Stony Brook helps the user to easily identify the location rather than having to examine the map to figure out where the location might be.

Reverse geocoding is the process of returning an estimated street address number as it relates to a given coordinate. For example, a user can click on a location in the map thus indirectly providing a coordinate and have information returned that reflects the estimated house number on that street. This house number is interpolated from a range assigned to that road segment. If the user clicks at the midpoint of a segment that starts with address 1 and ends with 100, the returned value will be somewhere near 50. We have to observe that reverse geocoding does not return actual addresses, only estimates of what should be there based on the predetermined range.

A **geocoder** is a piece of software or a (web) service that helps in this process.

In order to help the user we automatically display the reverse geocoded address when clicking any marker that represents an identified location. This reverse geocoded address is shown in two places.

- An Informational HTML window that opens when the marker is clicked that gives both the coordinates (latitude, longitude) and the reverse geocoded address inside the window.
- The address is also displayed on top of the map for the particular marker clicked.

We display the reverse geocoded address in the following format

- Estimated House Number
- Street
- City
- Zip Code
- Error range for the estimated house number

The below figure illustrates the reverse geocoded address being shown in both the html window and on top of the map.

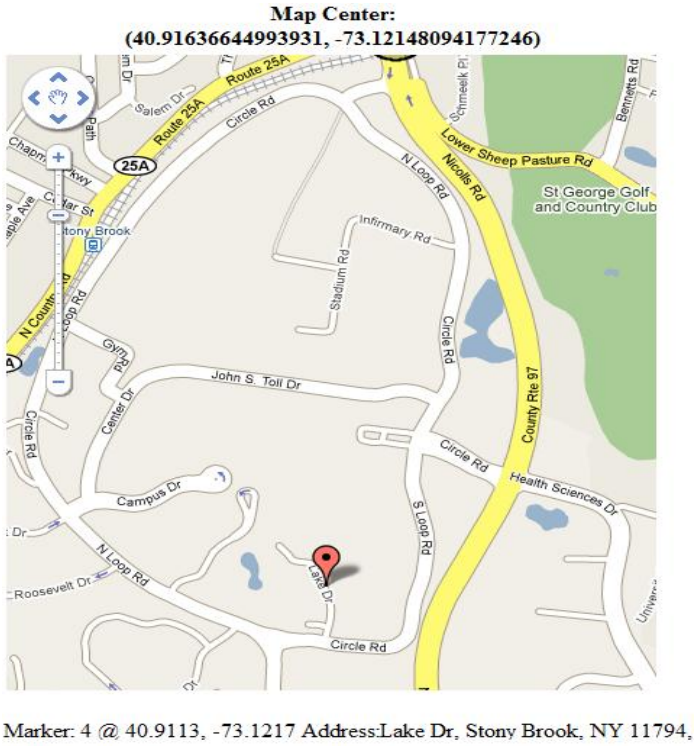

Figure 4.4: Reverse geocoding an activity location.

4.5.1.1 Implementing Reverse Geocoding

We make use of the Google Maps API [7] for reverse geocoding the latitude, longitude into the street address. The Google Maps API provides a client geocoder for geocoding addresses dynamically from user input.

We use the *GClientGeocoder* object to reverse geocode the location. The *GClientGeocoder* object supports the *getLocations()* method for reverse geocoding. We this method a *GLatLng* object representing the location, and then the geocoder performs a reverse lookup and returns a structured JSON object of the *closest addressable* location.

Because reverse geocoding is always an attempt to find the closest addressable location within a certain tolerance if no match is found, the geocoder will usually return a *G_GEO_UNKNOWN_ADDRESS* (602) status code. Once again the response is asynchronous in nature and so we supply an AJAX callback function which will be invoked when the response from the geocoder comes back to the client. Our

callback function then formats the response object and extracts relevant address fields and displays it on the map.

4.5.1.2 Improving Reverse Geocoding Performance

Reverse Geocoding is a costly operation in terms of network resources. It would need connections to the geocoder service every time when we want to display the reverse geocoded address. We optimize the reverse geocoding by building a client side cache that stores the reverse geocoded locations on the map. This is beneficial and it is applicable because of the following reasons:

- After clustering the number of known locations is fixed until we find a new unlabelled location as a part of new data collected.
- The address of these locations hence would remain constant after manual error correction if any is applied.

Hence building a client side cache avoids the need to establish network connections to the geocoder service to obtain the same redundant address information for a location.

Since reverse geocoding is an asynchronous operation it incurs noticeable latency. Caching the response invalidates the latency incurred and hence user responsiveness is improved.

We construct a client side cache that contains pre-computed geocoder responses to identified locations after training by first building an array of geocoder responses. Next, we create a custom cache that extends a standard GeocodeCache. Once the cache is defined, we call the setCache() method to use our cache.

4.5.1.3 Cache Revalidation on Marker Movement

Whenever the marker is moved we implement a standard cache revalidation policy. That marker is flagged as dirty and asynchronously in the background the new address is updated in the cache.

4.5.2 Street View

Google Street View is a feature of Google Maps [7] and Google Earth that provides for many streets in the world 360° horizontal and 290° vertical panoramic views from a row of positions along the street (one in every 10 or 20 meters, or so), from a height of about two meters. Google Street View displays photos taken from a fleet of cars. Each of these cars there are nine directional cameras for the 360° views, GPS units for positioning, Laser Range Finders for the measuring of buildings, and 3G/Wi-Fi aerials for whereabouts on 3G and Wi-Fi hotspots are all mounted.

Google Street View provides panoramic 360 degree views from designated roads throughout the coverage area for Google Maps. Google Street View uses the Flash® plug-in supported on most browsers to display these interactive images.

We show the street view images of a particular location if available in a small window next to our maps. This makes it extremely easier for users to recognize the locations by looking at the street view objects. Having the street view shown offers two distinct advantages to the labeling process:

- Labeling locations without an address becomes easier.
- There might be more than one place of activity that shares the same address. For example a grocery store and a pharmacy adjacent to each other might have the same reverse geocoded address but showing the street view helps the user to tag either one of these as the place of activity or both based on the context.

Figure 4.5: Street View shown to aid the labeling.

Chapter 5

Experiments and Analysis

5.1 Experiments

One of the unique features of our work is that it can use different sources for training. This provides a means to collect training data from a different device and directly integrate it into our framework to classify activities.

We collected data from drive traces that are collected automatically when the user drives the car during day to day activities. We show that we can use such data to train our framework to successfully analyze the activities of the user based on data collected at later stages from mobile phone. This is facilitated by making our framework transparent to the data source.

5.1.1 Data Source

We make use of drive traces collected for a span of about 23 weeks across 6 months. Data was collected for a total of 152 days with about total of 210 activities being performed during the collection phase. The geographical distance between the farthest points where activities occurred was about 54 miles.

Figure 5.1 shows the unique activity locations from the collected data on the map.

Figure 5.1: Activity locations from experiments.

5.2 Training with Collected Data

We trained the framework with 60% of the data collected. Activities were classified as **Stops** and **Transits**. A Stop is an activity that occurs for time $t > t_{delta}$ at the same location. In our analysis phase we set t_{delta} to be equal to 900 seconds or 15 minutes. A transit is defined as the connection element between two stops. We train to with all stop activities with respect to the car data based on the nature of data collected.

Training was done by running our clustering algorithm to identify unique activity points. Once we found the unique activities they were labeled with the help of tags.

At the completion of our training phase 182 activities were identified. Some of the activities were repeated every day while some activities were totally unique.

The framework also generated information about the number times each activity was performed and the average time spent with each activity.

5.3 Erroneous Data

It was identified during the training that some of the activity points were away from the ground truth location. This was because of the problem with the collected data during the drives. On careful analysis we identified three primary root causes that can cause such errors:

- Inherent GPS error because of hardware error in identifying the exact location. This error is usually small with a maximum of about 50 meters.
- Unable to obtain position fix at the activity location because the GPS was unable to lock on to the satellites without line of sight. This caused the GPS to report the previously identified fix as the new fix for the activity location. Such behavior was seen in locations like a Parking Garage where parking the car takes some time.
- Failing to flush in memory data to persistent storage before turning off The GPS writes the position updates to memory and if the file system is not synchronized some updates can be lost when the system is powered off. For Linux based systems (pdflush daemon of Linux VMM) in memory dirty pages are flushed once in 30 seconds only if there is no memory bottleneck. Since the drive traces were powered by car battery the system was powered off when the car was turned off without syncing the latest position update to the disk.

5.4 Representing Erroneous Data

When the training was run with data containing erroneous samples it resulted in more than one activity representation for the same activity. Such a representative can be visually identified by having a dense distribution of activity locations around the real activity location. Figure 5.2 shows 3 representatives being generated for single activity.

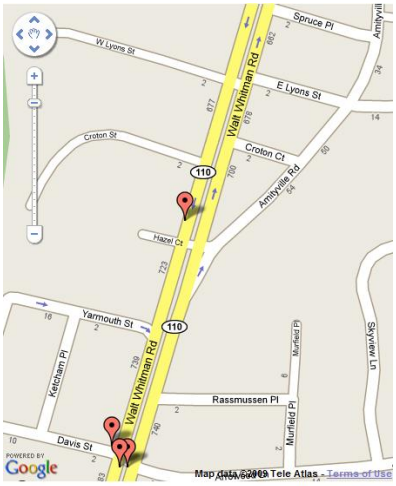

Figure 5.2: Erroneous Activity Locations.

5.4.1 Establishing Error Range

We would like to automatically figure out the maximum bound for the error from an incorrect data sample. While establishing such a bound is impossible for the error types (1) and (2) of section 5.3 we can figure out the error bound for erroneous data caused by failing to flush to disks.

For such errors we observe that the erroneous representative points might be valid members of some other drive trace for which the representative point is further down the street. This is because at some occasions the flushing to the disk happened with the most relevant location update. So these two drive traces would be identical except that the correct drive trace would have the incorrect representative point as an intermediate point in the trace.

Based on this observation we can arrive at a bound for the error range for such errors. Typically the distance of the erroneous data from ground truth depends on the distance covered by the car in 30 seconds. This is because once in 30 seconds we have the guarantee that the location update was flushed to the disk.

To compute the error range we calculate the velocity of the car from the last 20 samples of the drive trace and we calculate the maximum projection distance as

$$Error_{Projected} = Average\ decelerating\ Velocity/sec \times 30\ seconds$$

We pick the representative as the point with maximum number of neighbors where a point is treated as neighbor if it lies within the circle of radius Error projected around the point.

5.4.2 Representing Erroneous Locations

After we determine the representative point for the erroneous set of representatives and the maximum bound on the error range has been established we represent these points on the map with a different marker. The marker is shown with a circle representing the error with the radius of the circle equal to the error range and the center to be at the representative point. Figure 5.3 represents the error range for the activity location represented in Figure 5.2.

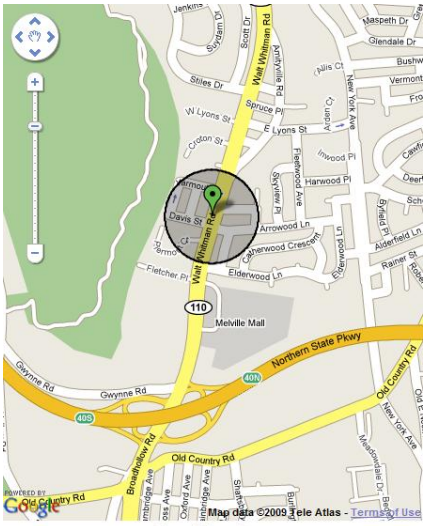

Figure 5.3: Representing Error Range and Erroneous Location.

5.5 Correcting with User Feedback

We have already observed that errors of type 1 and 2 cannot be automatically detected. While we successfully identify the type 3 error and show it on the map we would want the user to be able to correct the erroneous point despite the type of error. So we provide a user feedback based error correction into our framework to make this possible.

The mechanism of the error correction is very simple that the user can simply drag and drop the erroneous marker to the correct location. Such an event updates the location activity with the new location of the marker in our framework.

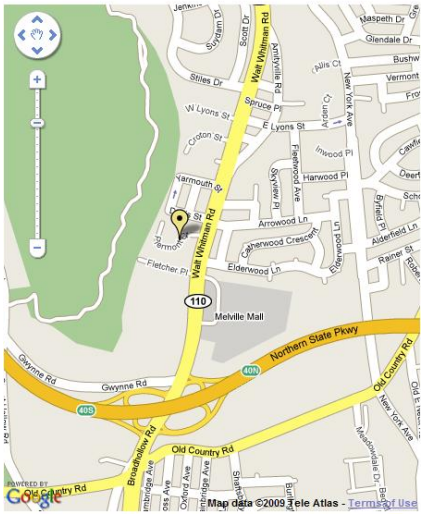

Figure 5.4: User Assisted Correction of Erroneous Points

5.6 Analysis Phase

After the training phase the user can choose any date for automatic analysis of activities. We provide the means to select the date by presenting a simple combo box of available dates. We analyze the activities on the day by matching it with learnt activities. The results are then pushed to Social networks or summarized to the story board as discussed in next Chapter.

Chapter 6

Integration with Social Networks

6.1 The Emergence of Social Networks

Social networking has encouraged new ways to communicate and share information. Social networking websites are being used regularly by millions of people. Typically a social network helps to establish a virtual online identity of the user that reflects various real life aspects of a person. The identity is commonly in the form of a profile created by the users for themselves. Today there are hundreds of social networking sites that cater to a wide variety of users, interests and practices.

A social networking site (SNS) can be defined [2] as web-based services that allow individuals to

- Construct a public or semi-public profile within a bounded system.
- Articulate a list of other users with whom they share a connection.
- View and traverse their list of connections and those made by others within the system. The nature and nomenclature of these connections may vary from site to site.

6.2 Types of Social Networks

Social networking sites can be broadly classified under two major types: Internal social networking (ISN) and external social networking (ESN) sites, such as MySpace, Facebook and Twitter. Both types can increase the feeling of community among people. An ISN is a closed/private community that consists of a group of people within a company, association, society, education provider and organization or even an "invite only" group created by a user in an ESN. An ESN is open/public and available to all web users to communicate and are designed to attract advertisers. ESN's can be smaller specialized communities (i.e. linked by a single common interest e.g. Great Cooks Community) or they can be large generic social networking sites (e.g. MySpace, Facebook etc).

6.3 Using Facebook as a Communication Platform

Social networking sites like Facebook present an incentive based mechanism as there is a social urgency to share information about oneself to others.

Modern social networking services allow for fine grained granularity in determining the recipients of shared information by employing privacy control mechanisms. Privacy control can be implemented in different levels according to social relations already available in Facebook.

We also gain a massive audience to try out our application as a result of the ubiquity of the social networks like Facebook.

6.4 Designing the Facebook Application

We design the Facebook application to present the results of the real time tracking application.

6.4.1 Structure of a Facebook Application

The Facebook Platform [6] help developers to create applications that interact with core Facebook features. The Facebook platform consists of three major components:

- API - The API defines the various methods through which you we interact with Facebook. The API is our main hook to make use of the social networking features of Facebook within our application
- FBML – Facebook Markup Language is a custom markup language based on various bits of HTML. It's similar to Coldfusion or ASP.NET's tag-based syntax, and is used to define the pages of our application.
- FQL – Facebook Query Language is SQL for Facebook. FQL is a powerful query language for which we use in situations where there are no existing helper methods in the API to accomplish our needs.

6.4.2 Developing our Facebook application

Facebook allows us to develop applications that can be integrated on the Facebook site that users can install as a part of the profile. We initially install the Facebook Developer application under our profile that helped us to develop our own

Facebook application. Besides helping in developing our application the Facebook developer application also provides features like monitoring the usage of our application and configures various settings pertaining to application distribution.

Facebook applications are hosted on an external web server that belongs to the developer. This allows the developer to write the application in any language or framework of their choice as long as the application can be hosted as a web service.

Our application is written in PHP that is hosted on a standard Ubuntu server machine running LAMP stack.

Figure 6.1 shows the screenshot of our Facebook application.

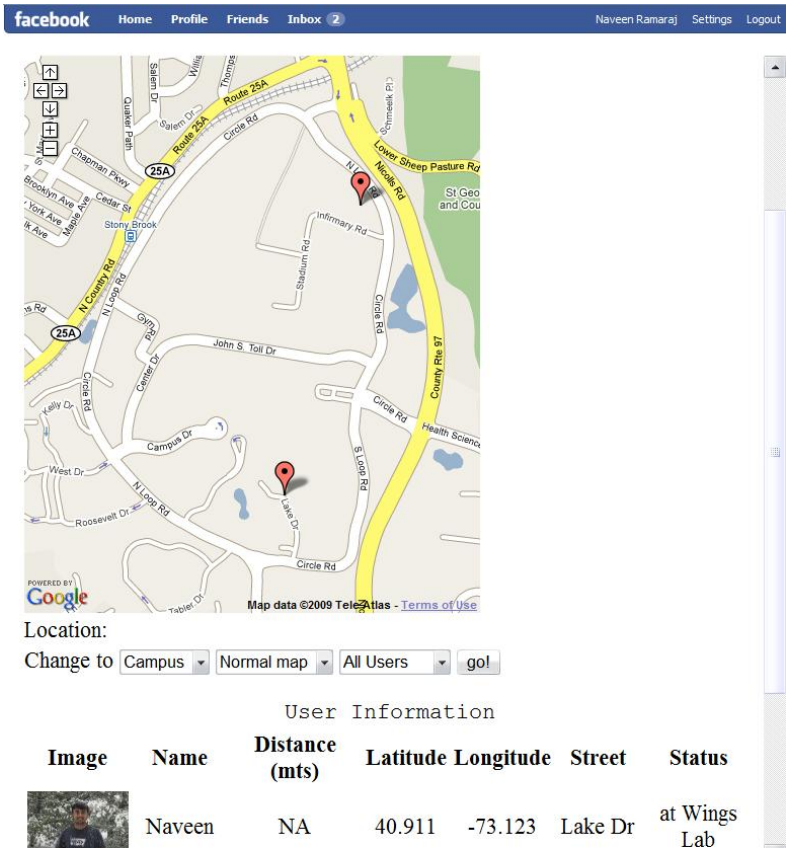

Figure 6.1: The Facebook tracking application.

6.5 Storyboard

Besides the tracking application we have developed a storyboard application to display the activity patterns in a time lined fashion. The idea of story board is to represent summarized information and to jump to any of the analyzed dates to view activities on that day. The storyboard events are dynamically generated by querying the learnt activities from the database. The visual representation of the story board makes use of Google Visualization API. Figure 6.2 shows the screenshot of our storyboard application.

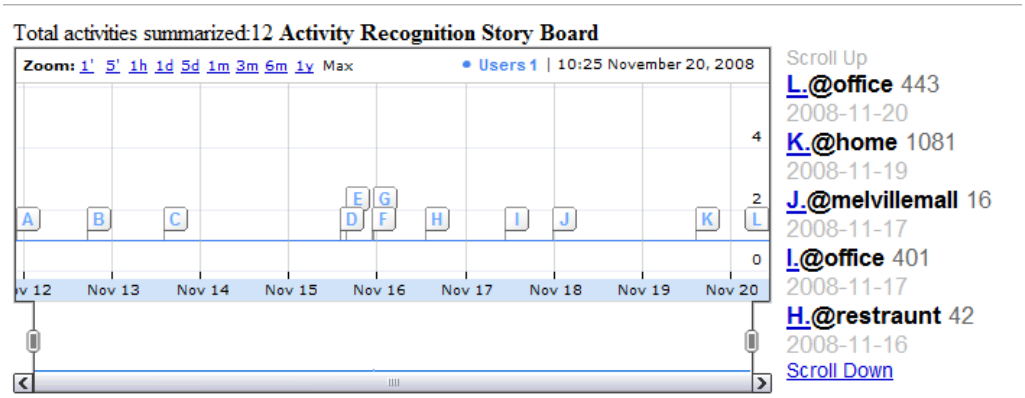

Figure 6.2: Storyboard application.

Each activity is represented by the description and also the duration of the activity. Clicking on an activity shows the custom attribute like a status message that was entered by the user. E.g: **@restraunt : "Having pizza".**

Chapter 7

Conclusion and Future Work

In our work we have studied the possibility of using mobile phones as a platform for activity recognition. In particular, we have proposed a framework to seamlessly obtain and process data from mobile devices to and mechanisms to predict activity recognition. We have solved the challenges associated with building the framework and we have also performed experiments and proved our concept of location driven activity recognition.

However the possibilities are endless in terms of physical activity recognition and using it seamlessly in all ubiquitous contexts. Some of the open challenges include predicting the exact physical activity by making use of other sensors like accelerometer and microphone or perform fine grained indoor localization using Wi-Fi fingerprinting or GSM localization.

Also there is a huge challenge to express the learnt activity patterns in natural language semantics to deliver real world summaries. This would need data mining techniques to mine the frequent patterns and also to identify slight deviations from most common activity pattern.

Finally there are more avenues to integrate the framework with existing services to provide information of a slightly different nature. One example would be to use the framework to receive location specific Twitter [21] (micro blogging) updates and so on. Such possibilities still remain an open challenge.

References

[1] 8motions, J2ME maps. http://www.8motions.com/.

[2] Boyd, d. m., & Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 13(1), article 11.

[3] T. Choudhury and A. Pentland. Sensing and Modeling Human Networks using the Sociometer. In Proc. of the International Conference on Wearable Computing, October 2003.

[4] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In Proceedings of SIGGRAPH '93, pages 135–142, New York, NY, USA, 1993. ACM Press.

[5] N. Eagle and A. Pentland. Social Network Computing. In Proc. of The Fifth International Conference of Ubiquitous Computing, pages 289–296, October 2003.

[6] Facebook, Facebook Developers. http://developers.facebook.com.

[7] Google, Google Maps API. http://code.google.com/apis/maps/.

[8] Gemmell, J., Bell, G., Lueder, R.: MyLifeBits: a Personal Database for Everything. Communications of the ACM 49(1), 88–95 (2006).

[9] IBM, IBM Websphere Micro Edition, www.ibm.com/software/wireless/weme/.

[10] Jcraft, Jzlib, www.jcraft.com/jzlib/.

[11] A. Kupper, G. Treu, and C. Linnhoff-Popien. TraX: A device-centric middleware framework for location-based services. IEEE Communications Magazine, 44(9):114–120, September 2006.

[12] Liao, L., Fox, D., Kautz, H.: Location-Based Activity Recognition using Relational Markov Networks. In: Proc of IJCAI-05, Edinburgh (August 2005).

[13] Leichtenstern K, Alexander De Luca A, Enrico Rukzio E (2005) Analysis of built-in mobile phone sensors for supporting interactions with the real world. PERMID, pp 31–34.

[14] Lester, J., et al.: A Hybrid Discriminative/generative Approach for Modeling Human Activities. In: Proc. of the 19th IntlJoint Conf. on Artificial Intelligence, Edinburgh, pp. 766–722 (2005).

[15] E. Miluzzo, et al. CenceMe - Injecting Sensing Presence into Social Networking Applications. EuroSSC, Kendal, Oct 2007.

[16] Morikawa, D.; Honjo, M.; Yamaguchi, A.; Nishiyama, S.; Ohashi, M., "Cell-Phone Based User Activity Recognition, Management and Utilization," Mobile Data Management, 2006. MDM 2006. 7th International Conference on , vol., no., pp. 51-51, 10-12 May 2006.

[17] C. Randall and H. Muller. Context awareness by analysis of accelerometer data. In Proceedings of the 4th. IEEE International Symposium on Wearable Computers, pages 175–176, 2000.

[18] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, et al. RFC 2616: Hypertext Transfer Protocol — HTTP/1.1, June 1999. http://www.ietf.org/rfc/rfc2616.txt.

[19] Stauffer, C.; Grimson, W.E.L., "Learning patterns of activity using real-time tracking," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.22, no.8, pp.747-757, Aug 2000.

[20]SUN, Java ME, java.sun.com/javame/index.jsp.

[21] Twitter, http://twitter.com.