Stony Brook University

Activity Recognition using WiFi Signatures

A THESIS PRESENTED

BY

RAJENDRAN THIRUPUGALSAMY

TO

THE GRADUATE SCHOOL

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

AUGUST 2010

Stony Brook University

The Graduate School

Rajendran Thirupugalsamy

We, the thesis committee for the above candidate for the

Master of Science degree,

hereby recommend acceptance of this thesis.

Professor Samir Das – Thesis Advisor,

Computer Science Department

Professor Jie Gao – Chairperson of Defense,

Computer Science Department

Professor Jennifer Wong – Committee Member,

Computer Science Department

This thesis is accepted by the Graduate School

Lawrence Martin

Dean of the Graduate School

Abstract of the Thesis

Activity Recognition using WiFi Signatures

by

Rajendran Thirupugalsamy

Master of Science

in

Computer Science

Stony Brook University

2010

Many high end mobile phones include a number of specialized (e.g., accelerometer, compass, GPS) and general purpose sensors (e.g., camera, microphone). The availability of these sensors has given rise to sensing applications in the areas of health care, gaming and social networks. Apart from its use in communication and localizing the WiFi "sensor" has not been used for much else. The presence of the WiFi sensor in high end mobile phones, laptops, netbooks, PDAs, etc., along with the widespread deployment of WiFi access points (APs) makes for a highly ubiquitous sensor. In this thesis we focus on how the WiFi sensor could be used for activity recognition. Our approach behind activity recognition is to observe a high level activity is indicated by a corresponding location, and WiFi signatures are a potential source of location information. Thus traces from WiFi sensors can be used to distinguish between locations and thus different activities.

We present the techniques devised and implemented for recognizing the activities of a user based on WiFi sensor traces collected from mobile phones. Our techniques are a combination of activity transition detection and activity classification techniques. The key idea behind activity transition detection technique is to quantify the similarity between WiFi scans and cluster them together, thus detecting the transitions in activities. The method uses a unsupervised algorithm based on scale space analysis to detect the transitions. Our approach behind activity classification is based on the repeating activity

patterns of an user. We use a supervised learning algorithm to classify the activities to one of the repeating activities labeled by the user.

We present the client-server system design of our prototype implementation. The sensing client runs in the mobile phones and captures the WiFi signature. The server consists of tools for activity transition detection and classification. Towards the end we provide metrics for transition detection and classification algorithms based on experiments and realworld user data.

Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

Many high end mobile phones [1, 2, 3] now come with a number of specialized (e.g., accelerometer, compass, GPS) and general purpose sensors (e.g., microphone, camera). The availability of these sensors has given rise to sensing applications in the areas of health care, gaming and social networks [5, 7, 8, 12]. Apart from its use in communication and localizing the WiFi "sensor" has not been used for much else. The presence of the WiFi sensor in high end mobile phones, laptops, netbooks, PDAs, etc., along with the widespread deployment of WiFi access points (APs) makes for a highly ubiquitous sensor. In this paper we focus on how the WiFi sensor could be used for activity recognition. The activities identified could be used as feeds to several applications, eg. making entries into a diary, automatic microblogging, updating status on social networking sites, etc.

Given the wide acceptance of WiFi technology, we are almost always in an environment where beacons from APs can be detected. These beacons captured by the WiFi sensor can be a rich source of information about our activities. Several activities that we perform can be uniquely identified by the "WiFi signatures" associated with them. Using WiFi signatures, we are able to identify activities like walking to school, attending lecture, having lunch at the food court, etc. with a high accuracy. We are also able to identify different activities within a house like, watching television, cooking dinner, sleeping, having breakfast, etc. Despite its ubiquitous nature and the potential of WiFi signatures as an important source of information; to the best of our knowledge the WiFi sensor has thus far not been studied by the research community.

In our context, a change in location indicates a change in activity. We follow a two step process to recognize activities.

1. The first step is to detect the boundaries of activities i.e., the start and stop times of an activity. Section 3 covers this step in more detail.

2. The second step is to classify the detected activity to a particular activity label. Section 4 covers this step in more detail.

Chapter 2

2. Related Work

There is growing interest in using mobile phones as research platforms for pervasive computing [11], [z13] and urban sensing [8], [7], [5] applications. The general approach can be broadly classified as active and passive. Active techniques use specialized hardware and software to achieve high precision localization. While passive techniques take advantage of signals from existing ubiquitous sensors such as WiFi APs.

A great deal of research effort has been devoted to indoor localization systems. Cricket [17] is a well known active localization system using ultrasonic and WiFi sensors. Place Lab [18] is a successful project where signals from different WiFi and GSM base stations are utilized for localization. UCSD's Active Campus project [19] assumes that the locations of the WiFi APs are known a priori. RADAR [20] also operates on WiFi fingerprinting, and is capable of achieving up to 5m accuracy in indoor settings.

However, not much research has been done in activity recognition using WiFi as a sensor. Most research has focused on camera, accelerometer [6], and GPS as a source of sensor data. [21] uses CRF models to extract significant places from GPS traces. SurroundSense [22] uses a combination of sensors (GPS, GSM, accelerometer) for context awareness. SoundSense [23] uses the diversity in sound frequencies for different environment for localization.

Chapter 3

3. Activity Transition detection

In this section we present techniques used for detecting the boundaries of activities from raw WiFi traces. The WiFi trace consists of a sequence of WiFi scans collected over a time period. A WiFi scan lists the various APs from which beacons have been received along with the signal strength (SS) in dBm of the beacons. So each scan *i* consists of a vector of (APi, SSi) records.

In [10], a similarity based technique is used to cluster digital photos captured over a period of time to "events" such as birthday parties, vacation, etc. We apply the technique to cluster together similar WiFi scans and use this to detect the activity boundaries.

3.1 Similarity Matrix Embedding

First we need to create the similarity matrix from the WiFi traces. Similarity matrix represents the similarity between WiFi scans. Specifically the (i, j) element in the matrix quantifies the similarity between the i^{th} and j^{th} WiFi scans. We design the following similarity metric for this purpose.

3.1.1 Similarity metric

The similarity metric should be designed such that it gives a low value for WiFi scans representing nearby locations and gives high value for scans from distant locations. Such a metric can be obtained by converting the signal strength of the APs to distance values.

From the two-ray ground reflection model, the received power can be computed as:

$$Pr_d = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4}$$

Given the large number of APs that are found in WiFi scans it is impractical to accurately determine the transmit power Pt and the transmitter gain Gt for each AP. We assume that Pt and Gt are constant across all APs. Hence we have:

$dK = \frac{1}{P_{r(}(d)^{1/4}}$, where K is a constant

Our WiFi scan logs report received power in dBm which we convert to mW and then calculate dK for each AP seen at a location. Hence at the location that a WiFi scan is performed we can get distance measures to all the APs that are in the scan. Using these distance measures from APs whose locations are unknown, we calculate the centered landmark distance coordinates as described in [9]. Let APi be the set of k APs visible at a location P and let Di be the distance from P to APi. Then the centered landmark-distance coordinates for P are:

$$C(P) = (D_1^2 - M, D_2^2 - M, \ldots D_k^2 - M)$$

where,

$$M = \frac{D_1^2 + D_2^2 + \cdots D_k^2}{k}$$

If C(P) and C(Q) are the centered landmark-distance coordinates for points P and Q then the Euclidean distance between C(P) and C(Q) is proportional to the actual Euclidean distance between P and Q. A similarity matrix can be visualized as shown in Figure 1. The (i, j) element of the matrix represents the difference between scans i and j. We compute the difference between two scans i and j to be the Euclidean distance between C(i) and C(j).

$$S(i, j) = |C(i) - C(j)|2$$

If scan i and j both result in the same set of APs being observed, then S(i,j) can be easily computed. If however the two scans have only a subset of APs that are common For example, if scan i results in set I = {AP1, AP2, AP3, AP4} and scan j results in set J = {AP1, AP2, AP3} then we assume that the signal strength of AP4 at location of scan j is equal to the noise floor which typically is -95 dBm. This value is used to determine a value for the distance between location of scan j and AP4. If $I \cap J = \phi$, then we assume the difference between scans i and j to be undefined.

3.1.2 Similarity Matrix

Using the similarity metric above similarity matrix is embedded. Figure 1 shows how to embed a matrix from the similarity values. Figure 2 shows a similarity matrix computed from experimental WiFi traces.

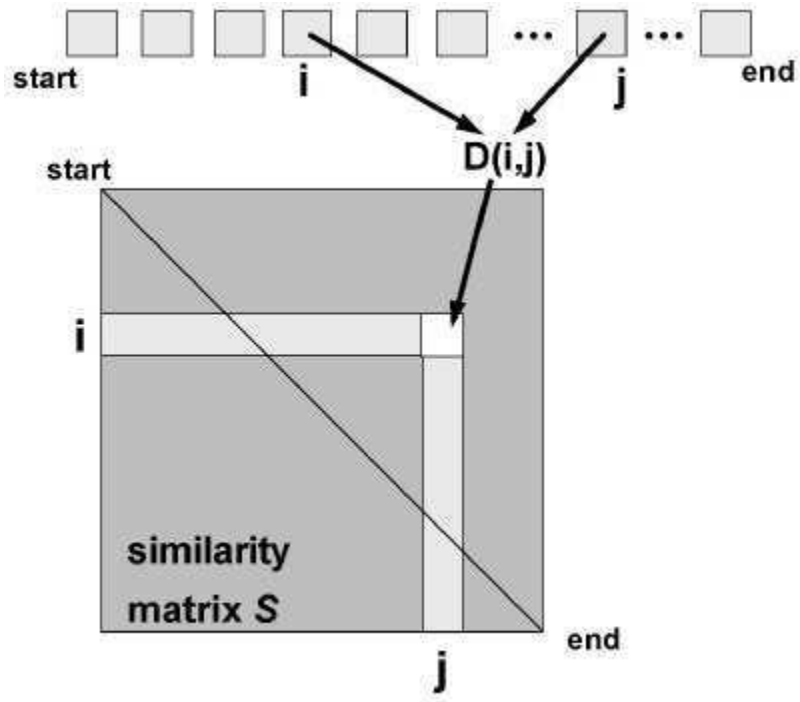

Figure 1: Embedding similarity data in a matrix

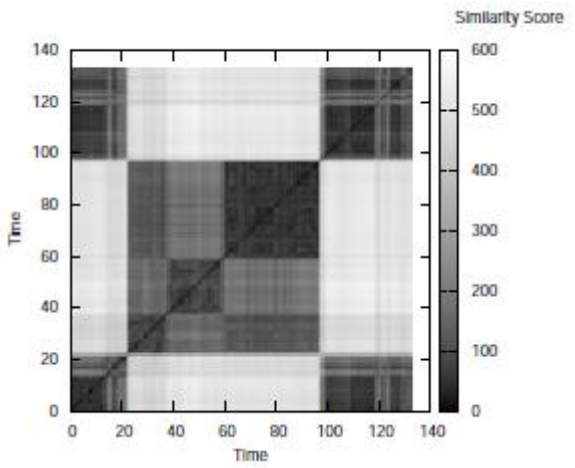

Figure 1: Similarity Matrix

3.2 Computing Novelty Scores

We can observe from the similarity matrix in Figure 2 that there are checkered squares along the diagonal. These squares indicate regions of small difference. To find the transitions between activities one has to simply find when one square ends and another begins. This can be done by designing a kernel as described in [10]. We describe the procedure here for completeness.

A unit kernel is a 2x2 kernel which can be decomposed into “coherence” and “anti-coherence” kernels as shown

$$\mathrm{K} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The first (coherence) term measures the self-similarity on either side of the center point of the kernel. The second (anti-coherence) term measures the cross-similarity between two regions. The difference of the two measures the novelty of the signal at the center point of the kernel. This will be high when the two regions are self similar but different from each other. Kernels larger than the unit kernel can easily be formed by taking the Kronecker product of K with a matrix of ones, e.g.,

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$$

Gaussian tapered kernels as shown below use windows that taper towards zero at the edges and as they are smoothed they avoid edge effects.

$$S_K(i,j) = \exp\left(-\frac{(i-j)^2}{2K^2}\right)$$

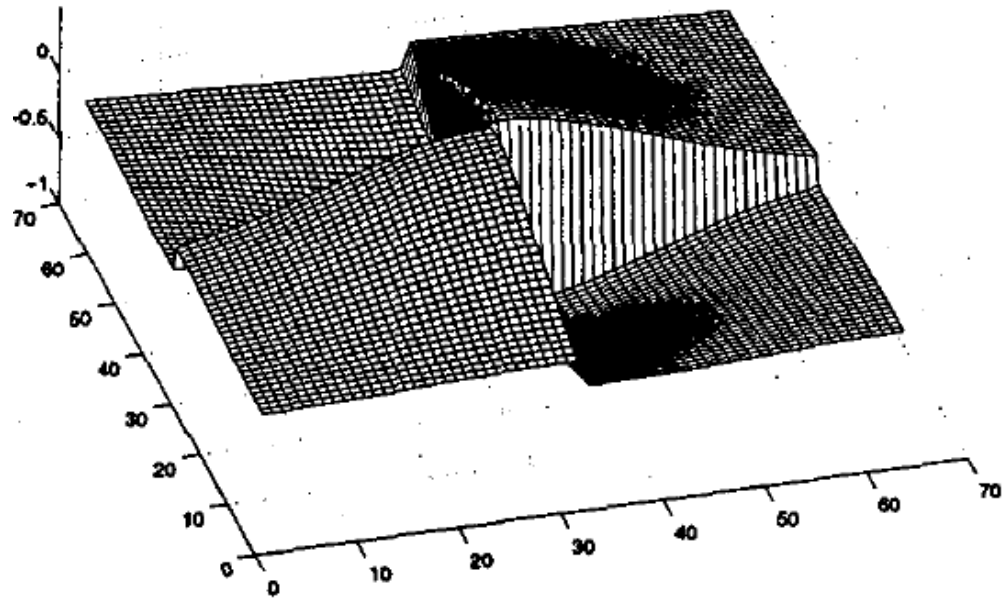

Figure 2: Checkerboard kernel

We can get a measure of novelty by correlating a kernel with the similarity matrix. We slide K along the diagonal of our similarity matrix S in Figure 2 and sum the element by-element product of K and S. When the center of K is at the point where one activity ends and another begins the overall sum will be large. Hence we get a time-aligned measure of the novelty N(i) of WiFi scans; where i is the scan number. The kernel is centered at 0,0 and has a lag of L. Figure 3 shows the novelty

score when a 4x4 kernel is moved along the diagonal of Figure 2. The threshold hits detect the transitions in activities.

$$N(i) = \sum_{m=-L/2}^{L/2} \sum_{n=-L/2}^{L/2} C(m,n)\, S(i+m, i+n)$$

The width of the kernel, L affects the novelty metric. If L is small then small transitions will be detected and when it is large the smaller transitions get averaged out and only larger transitions are detected. Figure 4 shows normalized novelty scores on Figure 2 using a 4x4 kernel and a 16x16 kernel. We can see that the 16x16 kernel produces a much smoother curve and smaller transitions that are detected by the 4x4 kernel are not detected using the 16x16 kernel.

3.3 Unsupervised transition detection

Due to the erratic nature of wireless signals a straightforward threshold based peak detection algorithm will detect lot of spurious transitions. For example, Figure 3 shows novelty score graph for a real world scan. Hence unsupervised leaning algorithms are preferred to detect the transitions. In this section we present an unsupervised algorithm based on scale space analysis for automatically detecting the activity transitions.

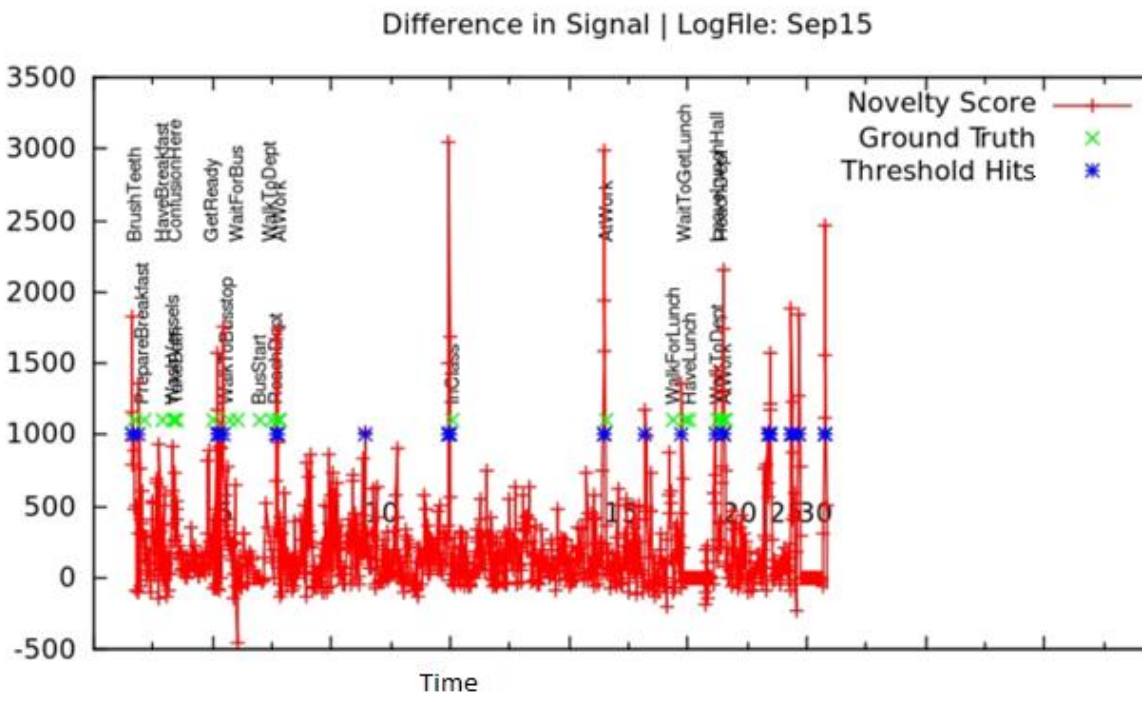

Figure 3: Novelty score graph for WiFi traces collected from real world user activities

3.3.1 Scale Space Analysis

Scale-space analysis [15] is a technique that can be used for finding peaks in a signal. We use this technique to detect activity transitions from the novelty score graph in an unsupervised manner. Novelty score dataset is applied to Gaussian filters of varying standard deviation to generate a set of scaled novelty score values. Each set is analyzed to detect the transitions at that scale. Standard deviation of the gaussian filter represents the scale and is used as a mean to expose the transitions at that scale and suppress other fine grained transitions below that scale. So the basic idea is to detect high level activity boundaries first and then slowly move onto detect more granular activities.

The space scaled novelty score dataset $N_\sigma(i)$ is calculated as follows,

$$N_S(i) = \frac{1}{2\pi\sigma^2} \sum_{j=-L}^{L} N(i) e^{-\frac{j^2}{2\sigma^2}} \quad (1)$$

where σ is the standard deviation of the filter and *N(i)* represents the novelty score dataset.

The following figure illustrates the effect of varying the σ parameter.

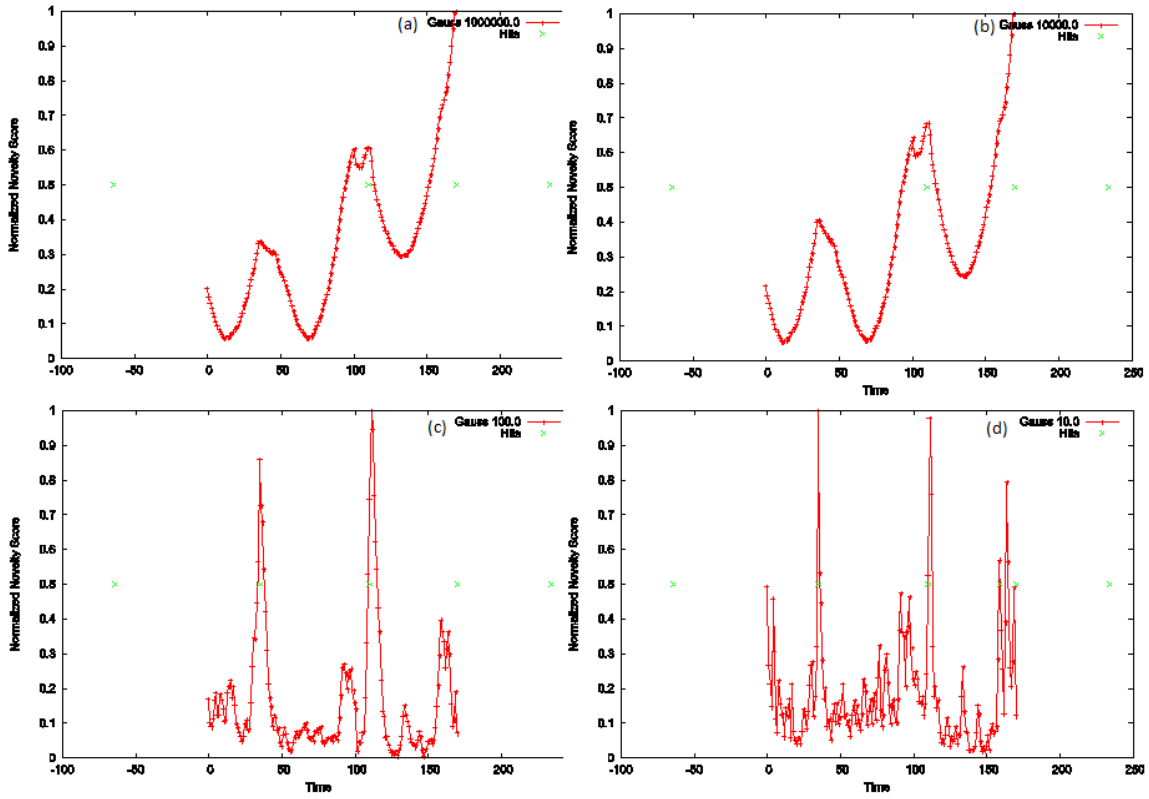

Figure 4: Novelty score graphs after applying different Gaussian filter
(a) K=1,000,000 (b) K=10,000 (c) K=100 (d) K=10

The following table describes the straightforward algorithm that detects peaks using scale space analysis.

```
Input:   Novelty score dataset {N1, N2, ... NT}

Output:  List of transitions

Transitions = {}
For Each σ in descending order
    Compute Nσ(i) from (1)
    peaks = Detect peaks in Nσ(i) that are above a percentage
threshold
    Transitions = Transitions U peaks
```

```
End For
Return Transitions
```

Table 1: Transition detection algorithm based on scale space analysis

3.3.2 Goodness Criteria

We need a criterion to assess the quality of the transitions selected in each iteration of scale space analysis and when to stop scaling down further. We use the similarity based confidence score for this purpose. It is defined as the sum of average intra-cluster similarity minus inter-cluster dissimilarity. If the space scaled novelty score dataset is $N_\sigma(i)$ and the boundaries detected for this dataset are $B_\sigma = \{b_1, \dots b_n\}$, the confidence score is defined as,

$$C_S(B_\sigma) = \sum_{l=1}^{|B_\sigma - 1|} \sum_{i,j=b_l}^{b_{l+1}} \frac{S_\sigma(i,j)}{(b_{l+1} - b_l)^2} - \sum_{l=1}^{|B_\sigma - 2|} \sum_{i=b_l}^{b_{l+1}} \sum_{j=b_{l+1}}^{b_{l+2}} \frac{S_\sigma(i,j)}{(b_{l+1} - b_l)(b_{l+2} - b_{l+1})} \quad (2)$$

The first term represents the average intra-cluster similarity and the second term the average inter-cluster similarity. The following figure illustrated this.

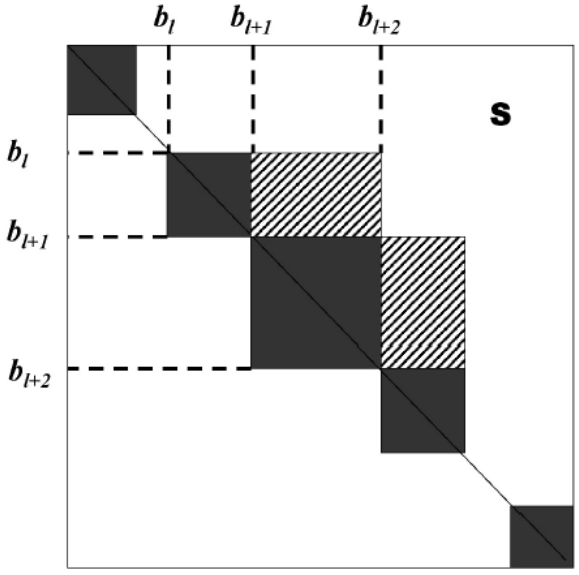

Figure 5: Confidence score computation:
Dark regions represent inter-cluster and grey regions represent intra-clusters

3.4 Summary

The following table describes the complete unsupervised algorithm for transition detection using scale space analysis and also utilizing the goodness criteria to stop further space scaling.

```
Input:   Novelty score dataset {N1, N2, ... NT}

Output:  List of transitions

Transitions = {}
For Each σ in descending order
    Compute Nσ(i) from (1)
    peaks = Detect peaks in Nσ(i) that are above a percentage
threshold
    current_goodness = Goodness_criteria from (2)
    If current_goodness > prev_goodness
        Transitions = Transitions U peaks
    Else
        Return Transitions
    End If
End For
```

Table 2: Unsupervised transition detection algorithm based on scale space analysis using goodness criteria

Chapter 4

4. Activity classification

In this section we present the design of the activity classification component of our system. The key idea behind our approach to classification is to observe that there is a pattern in a person's everyday activities. People often tend to have a lot of activities recurring frequently, and a small number of infrequent activities. So if the system is able to automatically correctly classify the frequent activities, the involvement of user is greatly reduced. Our system is based on this. It correctly classifies frequent activities learnt earlier and requires the users themselves to label other activities.

Our system maintains a training database with activities performed by the user earlier and the correct activity label provided by the user. During the real-time classification of activities, the system classifies the activities to one of the activities in the training database. The user reviews the activity labeling and either corrects incorrectly classified activities or creates labels for new activities. The system then saves these activities into its training database. And when the same activity is repeated again the system will be able to classify correctly. Thus the system learns new activities every time and will be able to correctly classify majority of the activities after some initial period of learning.

4.1 Single Activity classification

In this section, we will look at how a single activity is classified and labeled. A single activity is represented as a sequence of WiFi scans that are clustered together by the transition detection technique. We need to classify this to a single activity label. We follow the following simple method of classification.

1. Separately classify each WiFi scan to an activity label. We use a Naïve Bayes based classifier that takes a single WiFi scan as input and outputs the most appropriate activity label. Section 4.2 covers more about the classifier.

2. Find the histogram of the activity labels. The histogram counts the occurrence of activity labels.

3. Choose the activity label that has the highest frequency in the histogram, and output as the classified activity label.

4.2 Naïve Bayes classifier

Naïve Bayes is a simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions. Independence assumptions among the features imply that one feature is not related to another feature. In our case the signal strength from different APs constitute different features and they are independent. Signal strength from one AP is not related to signal strength from another AP. Due to this independence assumption, Naïve Bayes classifiers can be trained efficiently and are well suited when the total no of features is very high.

In our scenario, the total number of APs visited by a person increases as the person visits new locations, and will be a high value in the order of a few hundreds. So the dimensionality of the inputs is high, and Naïve Bayes classifier makes an optimum choice for our scenario.

Chapter 5

5. System Architecture

In this section, we present the system architecture of the prototype implementation. The system is client-server based. It has a client application running in the mobile phones, and a backend component running in server machines.

The following Figure illustrates the various components of the system.

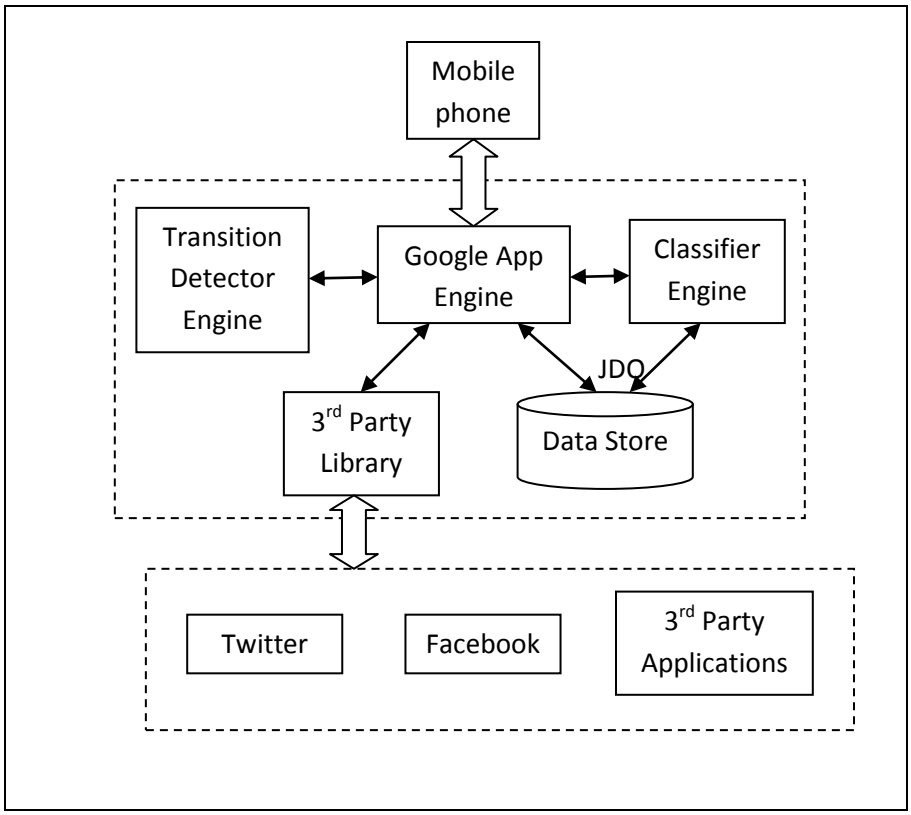

Figure 6: Software architecture of the system

5.1 Mobile Client

The mobile phone client application is implemented in C, and performs the following operations:

1. Scan for WiFi APs and log the WiFi signature of the current location. The scanning frequency is configurable by user.

2. Upload the logs to backend server for processing.

3. Show the processed results to the user and obtain changes in activity labels.

4. Upload the user reviewed activity labels to backend server.

5.2 Backend Server

The server component is written in Java and is based on Google App Engine. Google app engine [12] is a server platform for web applications and allows the code to run in Google's cloud computing infrastructure. It is similar to Apache/Tomcat web application servers. The server module has the following components:

1. **Google App Engine**: This is the core module that interfaces the mobile client with the other server modules. This component receives WiFi traces from the client and invokes corresponding modules to detect transitions and classify them. Then, it forwards the results to the client. And when the client gives the corrected transition labels it stores them in the datastore. Also this module interacts with 3rd party libraries to provide activity updates.

2. **Transition detector engine**: This component detects the boundaries of activities. The implementation is based on scale space analysis and goodness criteria as shown in Table 1.

3. **Classifier engine**: This component is implemented using Weka [24] and runs the Naïve Bayes classifier.

4. **Datastore**: This stores the training database for classifier and the WiFi traces of the user.

5. **External libraries**: Other 3rd party library components can be used to integrate the system with other applications, especially social networking applications such as Twitter [13], Facebook [14]. Our system implementation uses Twitter4J [15] library to post activity update tweets to Twitter.

Chapter 6

6. Evaluation

In this section we present the analysis of our system with experimental data and real world data collected from users. We separately provide evaluation results for the activity transition detection technique and activity classification technique. Our evaluation system is developed on the Openmoko [4] Neo FreeRunner as the mobile phone and a server machine for the backend server processing.

6.1 Activity transition detection

The phone is carried on a subject's body as he goes about doing his daily activities. The application scans the WiFi interface periodically and also provides the user with a drop down menu to choose from several activities that he is performing. Every time the subject makes a transition from one activity to another he indicates the change using the drop down menu. In this manner we maintain the ground truth. We had one user collect data in 2 environments. The first environment is a small 2 bedroom apartment and the other is an office building. To hasten the data collection process, the subject moved about the two environments pretending to do his daily activities. The phone's WiFi interface is scanned every 5 seconds. In total the subject had 84 activity transitions in the apartment and 78 transitions in the office building. In the apartment the phone detected 98 transitions out of which71 transitions matched the ground truth. In the much larger office building, the phone detected 83 transitions out of which 76 transitions matched the ground truth.

The precision, recall and F-score for the detected transitions appear in Table 1 for each environment. These measures are common figures of merit in information retrieval. Precision indicates the proportion of falsely detected transitions:

$$precision = \frac{\text{correctly detected transitions}}{\text{total number of transitions detected}}$$

Recall measures the proportion of true transitions detected:

$$recall = \frac{\text{correctly detected transitions}}{\text{total number of ground truth transitions}}$$

The F-score is a composite of precision and recall:

$$\text{F} - \text{score} = \frac{2 \text{ x precision x recall}}{precision + recall}$$

	Precision	Recall	F-score
Apartment	0.72	0.85	0.78
OfficeBuilding	0.92	0.97	0.94

Table 3: Precision, Recall and F-score for 2 environments

Figure 7: Openmoko phone running logging application.

Chapter 7

7. Conclusion and Future Extensions

In this thesis we presented the design, implementation, evaluation of a system for recognizing daily activities of a user from WiFi traces collected from mobile phones. The technique involves detecting the activity boundaries using similarity based clustering technique and classifying the activity using Naïve Bayes based classifiers.

Some of the limitations of the current system design and implementation and scope for future work include:

- Our system is passive in the sense that it requires user initiation to process the logs, detect and classify activities. The system can be modified to do real-time detection and classification of activities.
- The application of advanced algorithms for classifying activities could be explored.
- Integrating our system with more popular social networking utilities like Facebook, iGoogle, Google Buzz.

Bibiliography

[1] Apple iphone. 2009. http://www.apple.com/iphone.

[2] Google Android. http://code.google.com/android.

[3] Nokia N95. 2009. https://www.nokiausa.com/findproducts/phones/nokia-n95.

[4] Openmoko. 2009. http://www.openmoko.com.

[5] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. Pervasive Computing, IEEE, 6(2):20–29, 2007.

[6] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data, 2004.

[7] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications, pages 117–134, October 2006.

[8] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet, New York, NY, USA, 2006. ACM.

[9] Q. Fang, J. Gao, L. J. Guibas, V. Silva, and L. Zhang. Glider: Gradient landmark-based distributed routing for sensor networks. In in Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM, pages 339–350, 2005.

[10] J. Foote. Automatic audio segmentation using a measure of audio novelty. In Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on, volume 1, pages 452–455 vol.1, 2000.

[11] K. A. Li, T. Y. Sohn, S. Huang, and W. G. Griswold. Peopletones: a system for the detection and notification of buddy proximity on mobile phones. In MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services, pages 160–173, New York, NY, USA, 2008. ACM.

[12] Google App Engine. https://appengine.google.com.

[13] Twitter. http://twitter.com.

[14] Facebook. http://www.facebook.com.

[15] Twitter4J. http://twitter4j.org.

[16] D. J. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, and H. Kautz. Opportunity knocks: A system to provide cognitive assistance with transportation services. In UbiComp 2004: Ubiquitous Computing, volume 3205 of Lecture Notes in Computer Science, pages 433–450, Berlin / Heidelberg, 2004. Springer.

[17] N. B. Priyantha. The cricket indoor location system. PhD thesis, 2005.

[18] Y. Chen, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In ACM MobiSys, 2005.

[19] W. G. Griswold, P. Shanahan, S. W. Brown, R. Boyer, and M. Ratto. Activecampus - experiments in community-oriented ubiquitous computing. IEEE Computer, 2003.

[20] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In IEEE INFOCOM, 2000.

[21] L. Liao, D. Fox, and H. Kautz. Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields. in International Journal of Robotics Research, 2007.

[22] Martin Azizyan, Ionut Constandache, Romit Roy Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. in MobiCom 2009.

[23] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury and Andrew T. Campbell. SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones. in MobiSys 2009.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witter (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.