Context-Aware API for Android Devices: Outsourcing Report

Chris Smith
Supervisor: Naranker Dulay

January 2009

Contents

Part I

Background

Context- or Activity-Aware devices is an area currently under lots of research. There are many and varied applications of activity-aware devices, ranging from personal fitness and healthcare to training factory workers or merely playing music.

While this research is going on, there has been a huge expansion in the ownership, use and power of mobile telephones. Mobile telephones are so ubiquitous and now come with such a large sensor platform that they are the obvious choice for implementing activity-aware technologies for use in day-to-day life.

Making a context-aware API available on an open mobile platform such as Android will enable developers to start adding context-aware functionality to their applications without the extremely large overhead of writing a logger and classifier themselves, or re-engineering the application to use a context-aware framework if one is available.

1 Applications

There are many documented applications of activity-aware systems, and current research efforts which bring the technology to mobile telephones will only serve to lengthen this list.

The canonical example for activity-awareness, especially on mobile telephones, is modeling the user's "interruptibility"[30, 24]. This allows the software to know whether it's appropriate (or "polite") to disturb the user, and can advise the user's contacts when they are busy. It can also be used to create a "smart answering machine" [16] which can selectively direct calls straight to an answering machine if the user is engaged in an "uninterpretable" activity and the call does not appear to be important. These allow the user's mobile telephone to better approximate human behaviour - when approaching someone in person it is normally quite easy to determine whether it would be polite or necessary to disturb them, based on their demeanour, activity, and the urgency of your request; when picking up the telephone it is not possible at all without assistance from an activity-aware system.

The current implementations of these ideas have several problems, however. The more interesting research [16] requires a static camera fixed in an office to observe user behaviour, instead of implementing it directly on a telephone, which obviously constraints its usefulness. Of the two solutions actually targeted at mobile telephones, one[30] requires bulky custom hardware which the user must carry on their belt, and the other [24]does not expose an API to other applications and only surfaces the context-aware functionality in two small applications whose focus is on social interaction rather than improving the user's experience of the telephone locally. This project will aim to bring the ideas of these to generic hardware (an Android mobile telephone), and to provide an API which other applications can harness.

One use particularly suitable for mobile phones is dynamic adaptation of the device's settings based on the user's current activity and context[27]. When a user is walking the device can dynamically increase the font size to make it easier to read with an unsteady hand, and correspondingly decrease it when the user is stationary. In a similar fashion, the brightness of the backlight can be altered based on the ambient light level, and the ringer volume altered according to the noise level. Unfortunately this research did not progress beyond a feasibility study and was implemented on a Nokia 6110, which is severely outdated by today's standards.

Another popular area for activity-aware systems is in healthcare. Such systems can be used to monitor vulnerable people as they go about day to day activities to ensure that they're not in trouble - several systems[31, 20] can be used to monitor elderly persons and summon help if it is detected that they have fallen. Another healthcare application[33] allows nurses to remotely monitor the activities of their patients in a hospital ward, allowing them to respond to problems and keep up-to-date with their patients' well-being while not physically present. Activity-aware applications have also been used to try to encourage users to be more healthy; one novel application records the day-to-day fitness activities a user performs and uses this as a basis for a virtual "garden" that blossoms or wilts according to how much the user works out in a week[9].

As well as monitoring activities which the user is familiar with, activity-aware systems can also be used to assist users in learning new activities. One application[32] monitors the activities of trainee workers in a car manufacturing plant, and helps to provide a link between theoretical classroom-based training and practical work. The activity-aware system can offer advice to the workers that's directly related to the current task they're performing, and can even monitor their activities for compliance with procedures and give them a score afterwards.

While the research into healthcare and training applications present novel uses of activity-aware systems, the applications themselves are not really applicable to a mobile device or the scope of this project. The research does, however, describe the techniques used in those applications for activity classification and should prove useful in that respect.

Other areas of research include making activity-aware suggestions to the user [6], or issuing reminders or alerts based on the user's activity [26]. One example of the latter is an activity-aware system that detects when the user is making coffee, and plays a sound on a remote computer to alert thirsty coworkers to the fact. Sound isn't only limited to alerts, however: the XPOD[10] project is an activity-aware music player, which tailors the music being played to the user's current activity based on their past ratings. This type of activity-aware device presents a much greater level of personalisation than previously possible, and making this type of customisation available to mobile telephone users and application developers will surely result in many new applications.

2 Inferring activity

There are three general phases in most context-aware systems: a sensing component, which reads or receives raw sensor data relating to the user's environment

or activity; a feature extraction component, which analysis the sensor data and identifies a set of features from that data; and a classification component, which uses the extracted features to reason about the user's activity[8]. Each of these components will be expanded on below. Depending on the method of classification, some initial or continuous training may be required, and this is also considered below.

2.1 Sensors and devices

At the basis of activity-recognition are the physical hardware sensors. The most commonly used sensor is the accelerometer, which outputs the acceleration of the sensor[1] along a certain axis. There is extensive research on using accelerometers to classify activities such as walking [19, 12] (including whether or not the subject is walking on flat ground or up and down stairs [7]), running [12], falling [19], sitting [19, 12], cycling [12], etc.

Accelerometers can be combined with magnetic field sensors to convert the relative acceleration of the device into an absolute acceleration with respect to the Earth. This is particularly important when the sensor platform - a mobile telephone - can be oriented in different ways while performing the same activity. There are, for example, four different ways to comfortably fit a typical smartphone into a trouser pocket, made up of two possible orientations around two axis [2]. With just raw accelerometer data these four orientations would lead to different data for the same activity, but with the combination of magnetic field sensors we can normalise them.

Smartphones also come equipped with a microphone and GSM stack (prerequisites for a telephone conversation!), and commonly a camera, geolocation API (usually backed by GPS) and Bluetooth stack. With the exception of the latter two, these types of sensors are not particularly well explored for their use in context-aware systems at present. It is easy to reason how each would be useful - a microphone can reveal the ambient noise, which could indicate the difference between sitting in a library and a bar; the camera likewise can reveal the lighting conditions (if the device is not in a pocket or bag). The GSM stack can provide rough location information and also a signal strength to one or more cell towers; the signal strength will vary both with the user's proximity to the cell tower and the environment around them - being inside will degrade the signal more than being in open air, for example - so may provide vital clues to a context-aware system. One aspect of this project will be to research how the microphone, camera and GSM stack can be used to enhance existing activity classification algorithms.

Current research on location information and Bluetooth device proximity is summarised in section 4 (p12) and 5 (p12) respectively.

[1] and thus the device to which it's attached, and therefore the person using the device

[2] screen facing towards or away from the body, and the device oriented the right way up or upside down

2.2 Feature detection

It is not possible to reason directly about raw sensor inputs, so the next step in inferring activities is to extract useful *features* from the raw input. Features are usually mathematical properties of the input data, such as the difference between the minimum and maximum data point in a given time frame. Most classifiers use an extremely large number of features - [14] detect 562 different features from their inputs.

Some of the more commonly used features in activity-recognition systems are: mean, standard deviation, energy, entropy, correlation between axis, and discrete FFT coefficients[17]. Obviously, not all features are of equal value. FFT coefficients are generally very good indicators of activity, but the ideal coefficients and window sizes vary depending on the exact activity that is being detected. Likewise, the choice of other features to give the best recognition rate varies depending on the activity being detected[17].

As the sensor data is received continuously, it needs to be partitioned somehow before features are extracted. Most implementations use a sliding window approach with a 50% overlap between windows[4]. A window size of 10 seconds with a 50% overlap would result in one set of features being computed every 5 seconds. The window size is normally selected to correspond to a pre-defined number of samples to enable fast computation of certain features - most notably FFTs[4].

One challenge will be determining a set of features that are robust enough to perform activity analysis on, but are sufficiently inexpensive to calculate continually on a mobile device, where CPU speed is limited and excessive usage results in undesirable higher battery consumption.

2.3 Training

In order to meaningfully classify and label activities, some kind of training generally needs to be performed beforehand. The choice of classifier affects how much offline analysis has to be done on the training set, and whether or not it can be adapted at run-time.

One might expect that training would best be performed in a controlled environment, to reduce external influences on the user, but subjects in a laboratory setting are much more self-conscious about their movements, and this manifests itself in the data collected. Walking in a laboratory tends to produce acceleration data showing a consistent gait cycle which can be split into distinct phases, whereas walking in an uncontrolled setting produces data showing large fluctuations in gait phases and length. This means that classifiers trained on laboratory data may achieve a much lower accuracy when deployed in natural conditions[4].

2.4 Classification

The classification step involves feeding the features for frame into some kind of machine learning algorithm which can, using training data[3], determine which activity the feature-set most like represents. There are many different algorithms that can be used to perform the classification, some of which are discussed below.

2.4.1 Decision trees

Decision trees are possibly one of the simplest approaches possible[16]. A tree is constructed such that each node contains a test function, with branches for each possible discrete outcome of the function. This allows data to be classified with a "divide and conquer" approach. While high accuracy is possible in some circumstances[16], there are several drawbacks to decision trees: a plain decision tree has no way to model uncertainty - in an activity-aware system there will always be a degree of uncertainty as to the classification, and being able to measure this is an important tool. They also have an inductive bias which leads to a preference for the most general solution, and in most cases this generalisation causes many false results[29].

Decision trees require the structure of the tree and the test functions for each node to be determined during training. They do not lend themselves to minor on-the-fly modifications or new activities that are not part of the training set.

2.4.2 Neural networks

Neural networks are based on an extremely simplified model of the brain. The network consists of layers of neurons, and each neuron performs a simple arithmetic operation on its inputs. This normally consists of taking each of its inputs, multiplying it by a weight, and then summing all of the weighted inputs together; the resulting figure then becomes the neuron's output, and the input to one or more nodes in the next layer.

A network consists of a layer of input neurons, a layer containing one or more output neurons, and one or more layers of "hidden" neurons in between. The number of "hidden" layers, and the number of neurons within those layers must be chosen before training of the network begins. The training process will then determine the weights for each link in the network. The choice of number of layers poses a problem when designing a network, as too small a number can cripple the power of the network, but too large can cause it to be too expensive to evaluate and can possibly lead to it memorising erroneous data[10].

Neural networks, however, do provide good accuracy and could potentially (although not easily) be modified on-the-fly to cope with new activities.

2.4.3 Genetic algorithms

Genetic algorithms use the principle of natural selection to 'evolve' a solution to a problem. A set of random solutions are created, and a pre-defined fitness

[3] and any offline analysis made of that data

function is used to determine their relative worth. The best solutions are then combined together to produce the next generation of solutions, in a manner roughly analogous to reproduction in animals. Small "mutations" are also introduced into each generation to counter the effect of local maxima being reached.

Genetic algorithms can be combined with other techniques such as neural networks - the weights in the neural network can be "evolved" using genetic algorithms to create a neural network which is good as satisfying the fitness function.

The drawback of genetic algorithms is the need for a fitness function - the network will only ever be as good as the fitness function, and if you have a way to define what makes a good network you could hardcode

2.4.4 Instance-based learning

Instance-based learning (IBL)[35] algorithms are a class of "lazy" algorithms. They perform classification based on previously observed instances that have already been classified. There is no training required for IBLs, they're extremely adept at adapting to new scenarios, and they have a very low error rate[10] which makes them ideal for activity-recognition.

One particular type of IBL algorithm which is frequently seen in activity-aware research is the K-Nearest Neighbour (KNN) algorithm[13]. With the KNN algorithm, each sample is treated as a vector, and the distance[4] between the sample and the existing instances is calculated. The sample is then classified according to the classification of the majority of its k nearest neighbours.

One drawback of IBLs is that each new instance tends to be remembered for future use, which eventually results in large amounts of memory consumption and complexity when comparing distances of new samples. This can be partially overcome by only storing instances which would affect the classification of new samples[35].

The KNN algorithm can be easily extended to support dynamic classification of new types of activities - if a sample is not within a certain distance of sufficient other samples, it can be classified as a new type of activity.

2.4.5 Conclusion

There are numerous machine learning algorithms available and suitable for use in activity classification tasks. There has been a lot of research into their use, and all of the algorithms discussed have produced good results. Because of the lack of need for any training, however, the K-Nearest Neighbour algorithm appears to be the most promising for a mobile device. Any algorithm that needs explicit training prior to classification would almost certainly require a desktop application or a remote service to analyse the data, as it typically requires large amounts of memory and expensive computations. This either makes the application extremely cumbersome for the user (they have to connect their phone to a computer, transfer a file, obtain and run a separate application,

[4] the euclidean distance is usually used, but any metric will suffice

then transfer some file back), or puts a large resource burden onto the distributor (having to remotely analyse all of the data from all users would require dedicated hardware for any more than a few users).

3 Mobile telephones

It's hard to overstate the ubiquity of mobile telephones at present. In 2003, over a billion mobile telephones were sold - six times as many as the number of personal computers[11]. In 2007, this same figure describes the number of cameraphones sold[25], clearly representing a substantial growth in sales and advancements in the technology. In fact, mobile telephones are the fastest adopted technology in human history[11]. This ubiquity, coupled with the fact that mobile telephones are comfortably carried around on a daily basis by most of their users, makes them a very attractive alternative to more traditional platforms used for activity-aware research, which typically involved bulky or inconvenient apparatus that was expensive to manufacture[28] and made users very self-conscious.

3.1 iPhone

There have been several published works related to activity-recognition on the iPhone. The similarity between iPhone and Android platforms means that many of the concepts developed for or used on the iPhone are applicable to both.

3.1.1 iLearn

iLearn[28] is a suite of three tools - iLog, iModel, and iClassify - which together allow for real-time classification of low-level activities. iLog is run on the user's iPhone and allows the user to specify which activity they will be performing. The application then records raw sensor data from the iPhone's three-axis accelerometer and 124 features computed from this data in real-time. The data is then stored on the device, annotated with the selected activity.

The training data collected by iLog is then transferred to a desktop computer where iModel uses a Naïve Bayesian Network (NBN) to create a model which can be used to classify future input. The choice of NBNs was based on their ability to classify a set of trial data correctly, and the low computational cost of classifying data once the model has been generated.

Once the model has been created, it is transferred back to the device where it is used by iClassify. This provides an API for other applications, and allows them to register for a callback which it publishes the user's current activity to every second.

Unfortunately, neither the source code nor the API are published. The inability to run background processes on the iPhone suggests that any "API" would have to be more like a framework where the third-party developer has to re-engineer their application to use the iClassify application as a base. This is undesirable as it makes it extremely difficult to adapt existing applications to use the

activity-aware API, and is a very cumbersome way of providing what could be a very minor piece of functionality for the application.

3.1.2 Evaluation

[21] present an evaluation of the iPhone for use in "people-centric sensing applications". One of the major drawbacks highlighted is that the iPhone does not support applications which run in the background. This means that any application wishing to perform continuous real-time activity detection would need to run as a foreground process, preventing the user from using the device for any other function.

The research also shows that the computational compatibility of the iPhone is more than sufficient to perform the necessary calculations for a typical activity-recognising application, which suggests that any modern smart phone would be capable of these.

3.2 Android

While the Android platform is relatively new, it is rapidly gaining market share on the more established mobile operating systems. A December 2009 survey[1] shows that 21% of respondents want their next smartphone purchase to run Android, a 350% increase from the same survey conducted three months prior. This is compared to the iPhone, which dropped 4% to 28% in the same time period. Gartner, a respected IT research firm, predicts that by 2012, Android will be the second most popular mobile operating system globally[2].

In addition to its rapidly increasing popularity, the Android platform offers several advantages over the iPhone platform. Most notably is the ability to run background processes (called SERVICES), which will allow a classifier application to run without interfering with the user's normal use of their mobile telephone. In addition, the Android OS provides access to the Bluetooth and GSM stacks, allowing for data from both to be used for activity detection.

The ability to run a background process will enable a proper API for sharing activity data with other applications, which will allow third-party developers to make their applications context-aware with relatively little work on their part. This is extremely desirable as it will allow rapid prototyping of applications, which will hopefully lead to innovative new uses of activity classification.

While it is purported[12] that there is research being done on bringing activity-awareness to Android platforms, there does not seem to be any work published on this matter or any applications available to support it. While there a small number of self-proclaimed "context-aware" applications for Android, this context is almost exclusively limited to geolocation. This project will therefore produce one of the first publicly available activity-aware applications for the Android platform.

4 Location analysis

Location-based services are currently undergoing an "explosion"[5], thanks to improvements in technology, and greater openness on the part of service providers and handset manufacturers. All modern smartphone platforms have a geolocation stack, usually backed by a GPS chipset and in most cases augmented with either a database of known cell tower locations, or a map of known wifi network identifiers and locations, or both. The two databases allow for rough geolocation when GPS is not available, or for greatly decreased lookup time when a GPS lock is available.

However, while the geolocation stack is a rich source of data, it is a poor source of information. A latitude/longitude pair may describe the user's exact location, but a user would be hard-pressed to tell the difference between the latitude/longitude of their home, place of work, or of somewhere in between the two with no real significance. A great deal of research has therefore been devoted to detecting meaningful locations from GPS traces.

"Place recognition" has two phases: learning and recognising. An initial learning phase analyses a sensor log and segments the data into places where the device is stable (stationary), and designates this as a "waypoint". It then merges "waypoints" that appear to identify the same place being visited multiple times. The second phase uses these learned waypoints to recognise when the device is revisiting a place, and therefore also when the device is not visiting a previously known place (for example when it is moving between two)[15].

Unfortunately, quite a lot of research into location analysis uses GPS "blackspots" to identify useful places[23, 18]. With older GPS chipsets, the satellite signal would be lost when the user entered a building, and this allowed an inference that the current location was probably a place of interest. However, modern GPS chipsets receive a signal in most indoor locations. It is possible that a decrease in signal strength or number of locked satellites may still occur, or that GSM signal strength could be used instead, but these ideas have not been widely explored at present.

There is, however, plenty of research relating to the use of location data outdoors. One application[18] learns not only the user's frequently visited places, but the method of transport used between them and the typical routes taken. It can then offer instructions showing the user how to go from place to place, or issue alerts if the user appears to be going the wrong way (by getting on the wrong bus, for instance). The ability to correctly infer the user's destination would be extremely useful in a context-aware system: a user walking to do their grocery shopping is almost certainly going to want to interact with their phone differently than a user on a bus going to work.

5 Bluetooth

The user's context depends on not only what they are doing, where they are doing it, but also who they are with. Sitting and eating lunch with a manager is quite a different context to sitting and eating lunch with a spouse. It

would therefore be desirable to be able to identify between different people when performing context analysis.

One of the few ways that a mobile telephone can identify other people is by searching for *their* mobile telephones. This can be done by scanning for Bluetooth devices, which involves broadcasting a "device inquiry" message; if a device chooses to answer the inquiry, it discloses its unique MAC address and device class[5]. Unfortunately, this requires the person to not only be carrying a mobile telephone, but a Bluetooth-enabled model, and for them to have configured their device to have Bluetooth enabled and to be "visible". A study in 2004[11] found that only 1 in 150 people had such a configured device on a university campus. This figure will undoubtedly be greater now, and may well be greater when in public, but it highlights that only a handful of people may be detectable via their Bluetooth devices.

A study in 2006[22] used a similar technique to monitor the social context of the user, introducing the idea of "familiar" people, "unfamiliar" people and "familiar strangers". These labels were applied based on the number of times their Bluetooth devices were detected [6]. While the definition of "familiar" and "unfamiliar" are quite obvious, "familiar strangers" is a new class of people used to describe those who the user encounters repeatedly, but doesn't interact with. This may include neighbours that are passed on the street, or fellow commuters on a journey into work. The number of people in each of those groups (and any changes in those numbers) can be used to infer how "comfortable" the user feels with their social context, and whether their current activity is part of a normal routine or is novel.

This research has, to date, not been readily combined with activity-aware applications, and this project will aim to integrate the results of Bluetooth scanning with "classical" activity classification techniques and to evaluate whether it provides any benefit.

6 Power management

One major consideration when deploying an application on a mobile device is the amount of power it will use. An application constantly polling any one sensor can reduce battery life significantly, and an application which kept all available sensors active (in addition to doing CPU-heavy analysis on them) would drain the battery in a typical smartphone in a matter of hours. A context-aware application is not very useful for a user if they can only use their telephone for an hour or two before it needs recharging!

One solution[34]is to only use one or two sensors to monitor the user's activity until it appears to be transitioning. For example, if the user is believed to be walking, the application only needs to periodically check either the accelerometer (to confirm the user is still making walking motions) or GPS (to confirm the distance traveled is still consistent with walking) to know that their activity

[5]the device class tells us whether the device is a computer or a mobile telephone, for example

[6]and by extension the number of times the user had come into contact with them

has not changed. As soon as the user's behaviour becomes inconsistent with walking, the application can bring other sensors online until it has successfully reclassified the activity, and then resume monitoring with minimal sensors.

Another option[34] (which can be used in conjunction) is to only enable sensors for a short amount of time, and then sleep for a period before reactivating them. The "duty cycle" suggested for accelerometers is 6 second of sensing followed by 10 seconds of sleeping. The six second window is enough time to allow for capturing a full range of motion (several complete strides) if the user is walking or running, and then the ten second sleep stops the accelerometer using battery power until the next cycle. This process obviously means that a sudden switch in activity will not be noticed immediately, but a delay of a few seconds is acceptable as most activities will last for minutes or longer.

The battery life on modern smartphones rarely exceeds 24 hours of typical use, so it is extremely important that any applications developed for this project does not significantly reduce this. A balance between prompt detection and notification of activity changes and battery use by sensors and processing algorithms will need to be found.

Part II
Specification

7 Experimentation

7.1 Battery use

The average lifetime of a typical Android mobile telephone will need to be ascertained, taking into consideration typical day-to-day use of the device and any additional battery drain caused by that. The impact of sensor use on the battery lifetime needs to be calculated or experimentally determined, and the effect of the algorithms described in section 6 needs to be quantified.

Once this is complete, a selection of algorithms and suitable parameters should be made for use in the classifier application (section 8.1). The aim of the selection should be to ensure the best activity detection whilst not severely reducing the device's battery life.

7.2 Location analysis

As discussed in section 4, existing algorithms to detect interesting places rely on GPS signals not being obtainable indoors, which is no longer true with current generation GPS chipsets. It may be possible to adapt these algorithms to work instead using the GSM signal strength or number of locked GPS satellites. The variance of these when inside and outside a building need to be investigated to determine if this is a feasible approach, or whether an alternate algorithm needs to be used.

A suitable algorithm should be implemented which allows the user's activity to be annotated with location-based context. This would ideally take the form of a source and destination for mobile activities, or location for static activities. The goal is to be able to differentiate between "walking to work" and "walking to the shop", although the labeling of places may be left to future work. Unlabeled places (i.e., something akin to "walking to place2" and "walking to place7" instead of the previous examples) still allow for applications to tailor their behaviour based on the user's previous activities at or when walking to that place, so will be sufficient for an initial version.

7.3 Bluetooth usage

The algorithm for classifying familiar, unfamiliar, and "familiar stranger" Bluetooth devices (section 5) should be implemented. An experiment should then be conducted to determine whether or not the data provides useful clues as to the user's context. This will depend on the number of other people with Bluetooth discoverable devices in the vicinity of the user.

If there are a sufficient number of discoverable devices, then the algorithms should be included in the classifier application to provide additional context.

8 Deliverables

The following items should be delivered:

8.1 Context-aware Framework

An Android application which:

- monitors raw sensor data and extracts features from the device's:
- accelerometer sensors
- magnetic field sensors
- camera
- microphone
- monitors relevant data and extracts features concerning:
- visible Bluetooth devices in proximity to the device
- the current location of the device
- the current GSM signal strength and cell ID
- presents the user with a method of optionally annotating this data with their current activity
- classifies the user's current activity using the extracted features and a K-Nearest Neighbour algorithm
- enhances the classification with extra contextual data inferred from location or Bluetooth service:
- the user's probable source and destination, if traveling
- the user's current location, if stationary
- the user's company, if detected
- attempts to conserve battery life by intelligent management and timing of sensor activity

8.1.1 Interface

The application should provide two different interfaces to retrieve the user's context. The first should be an implementation of the Android ContentProvider interface. This allows third party applications to query and retrieve data (in this case, the user's probable activities and context) as and when they need it.

The ContentProvider interface exposes data as a table. The classifier application should therefore supply a unique ID for each possible activity, the name of the activity (if labeled), the time the activity was started, and the estimated

uncertainty in the classification. It should also expose relevant details of any context information, such as related places (s7.2) or other participants (s7.3).

The second interface should use Android broadcast Intents to notify any interested application whenever the user's activity changes. This will allow applications to be notified whenever the activity or context of the user changes. The Intent should contain a URI which corresponds to an entry in the table returned by the ContentProvider, so applications can retrieve information about the user's activity directly.

Whenever activities, places or participants are exposed in the interface, they should be assigned an ID which remains consistent for the specified activity, place or participant. Applications can use this ID regardless of any annotations the user supplies. For example, the first time an interesting place is detected it may be assigned ID 42 and label "place42", the user may then annotate this place as "home", and some time later replace it with "old house"; the ID will be consistent across all of these labels, so any application using this information will continue to function.

8.2 Activity condition for Locale

As a proof-of-concept, an application should be developed which provides Locale[3], a popular Android application used to alter device settings and perform actions based on certain conditions, with a new condition representing the user's activity.

This will allow users to change their device settings based on their current activity by creating rules within Locale. Locale also allows users to post updates to popular social networking sites such as Twitter and Facebook, so it will be possible for users to broadcast their activities to their friends.

8.3 Context-aware home screen

To demonstrate the utility of applications being context-aware, an application will be developed that replaces the standard Android home screen. It should dynamically adapt its layout and contents according to the user's activity and other context information made available by the classifier application.

The replacement home screen needs to allow users to browse and launch applications, and should expose relevant (context-sensitive) notifications to the user, including notifications of e-mails, missed calls, text messages and upcoming appointments.

8.4 User and developer guides

- A user guide for the classifier application
- A user guide for the Locale condition provider
- A user guide for the Context-aware home screen

- A developer's guide for using the information exposed by the CLASSIFIER application

8.5 Website

A website should be created containing an overview of the applications, instructions for both users and developers, and links to download the applications. The website should explain the basic ideas behind activity and context aware applications, and the specific techniques used in this project.

Part III

Evaluation

9 Reports

The results of the experimentation described in section 7 should be written up as a report. The reports must include the data collected in each of the experiments, the conclusions drawn from those, and the impact of the results of the experiment on the project deliverables.

10 Deliverables

On successful completion of the project, there should be three deliverable applications as specified in section 8. These can be tested and evaluated in a variety of manners:

10.1 Unit tests

Throughout the development of the project, unit tests should be created to test key functionality of all applications. It is expected that at the completion of the project, all unit tests should pass successfully, and they will have a code coverage of 80% or above.

10.2 System tests

The classifier application should also have a suite of system tests. These should consist of a set of fake or pre-recorded inputs which are fed into the application in place of raw sensor data. The output of the classifier (via the API) can then be compared to expected output for the data.

10.3 User acceptance testing

The Locale addon and context-aware home screen should be subject to user acceptance testing for evaluation. This should take the form of providing the applications to multiple end users, allowing them to use them for a period of time (providing instructions for certain tasks to complete). The users should then be presented with a questionnaire which they can use to evaluate the functionality, utility and design of the applications.

10.3.1 Android market

In addition to providing the applications to a closed set of users, the applications should be published to the Android market. This will allow any owner of a compatible Android device to download and use the applications. The market also allows users to rate the application and leave comments, which will be an extremely useful method of evaluating the success of the applications.

As part of this, the classifier application/framework will be published, and should include instructions for developers detailing how they can make their applications context-aware. The interest expressed and number of applications which use it (if any) will be an indicator of the effectiveness of the API and its documentation.

10.4 Classification scope

The classifier should be able to classify the following activities:

- walking
- running
- standing
- sitting
- traveling in a vehicle

These should be evaluated by means of a leave-one-out cross-validation test, with data collected from one or more users and annotated by hand. It is expected that the classifier should correctly classify all activities with an accuracy of at least 70%, within 30 seconds of the activity being started.

10.5 Resource usage

Finally, the applications should be evaluated based on their resource usage. One of the main concerns will be battery usage (caused both by the application using processor time, and activating sensors), but attention must also be paid to memory consumption (especially after extended use). The goal should be to ensure that the resource usage of the applications in this project do not adversely impact the functionality of the device; that is, battery life should not be reduced so significantly that it requires users to change their normal charging behaviour, and memory usage should not be so high as to cause other applications to become sluggish or be closed by the operating system.

References

[1] http://www.changewaveresearch.com/articles/2010/01/smart_phone_20100104.html.

[2] http://www.computerworld.com/s/article/9139026/.

[3] http://www.twofortyfouram.com/.

[4] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, 3001:1–17, 2004.

[5] P. Bellavista, A. Kupper, et al. Location-based services: Back to the future. 7(2):85–89, April–June 2008.

[6] V. Bellotti, B. Begole, et al. Activity-based serendipitous recommendations with the magitti mobile leisure guide. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 1157–1166. ACM, New York, NY, USA, 2008. ISBN 9781605580111.

[7] J. Carós, O. Chételat, et al. Very low complexity algorithm for ambulatory activity classification. In *European Medical & Biological Engineering Conference and IFMBE European Conference on Biomedical Engineering*. 2005.

[8] T. Choudhury, S. Consolvo, et al. The mobile sensing platform: An embedded activity recognition system. 7(2):32–41, April–June 2008.

[9] S. Consolvo, D. W. Mcdonald, et al. Activity sensing in the wild: a field trial of ubifit garden. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 1797–1806. ACM, New York, NY, USA, 2008. ISBN 9781605580111.

[10] S. Dornbush, K. Fisher, et al. Xpod - a human activity and emotion aware mobile music player. In *Proc. 2nd International Conference on Mobile Technology, Applications and Systems*, pp. 1–6. 2005.

[11] N. Eagle and A. Pentland. Mobile matchmaking: Proximity sensing and cueing, 2004.

[12] A. Garakani. *Real-Time Classification of Everyday Fitness Activities on Windows Mobile*. Master's thesis, University of Washington, 2009.

[13] J. Han and M. Kamber. *Data mining: concepts and techniques*, chap. 6, pp. 348–350. Morgan Kaufmann, 2006.

[14] A. Hein and T. Kirste. Towards recognizing abstract activities: An unsupervised approach, 2008.

[15] J. Hightower, S. Consolvo, et al. Learning and recognizing the places we go. In *UbiComp 2005: Ubiquitous Computing*, pp. 159–176. 2005.

[16] S. Hudson, J. Fogarty, et al. Predicting human interruptibility with sensors: a wizard of oz feasibility study. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 257–264. ACM Press, 2003.

[17] T. Huynh and B. Schiele. Analyzing features for activity recognition. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pp. 159–163. ACM, New York, NY, USA, 2005. ISBN 1-59593-304-2.

[18] L. Liao, D. J. Patterson, et al. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, April 2007.

[19] M. Mathie, B. Celler, et al. Classification of basic daily movements using a triaxial accelerometer. *Medical and Biological Engineering and Computing*, 42(5):679–687, September 2004.

[20] U. Maurer, A. Rowe, et al. ewatch: a wearable sensor and notification platform. In *Proc. International Workshop on Wearable and Implantable Body Sensor Networks BSN 2006*, pp. 4 pp.–145. 2006.

[21] E. Miluzzo, J. Oakley, et al. Evaluating the iphone as a mobile platform for people-centric sensing applications, 2009.

[22] T. Nicolai, N. Behrens, et al. Exploring social context with the wireless rope. In *In Proc. Workshop MONET: LNCS 4277*. 2006.

[23] P. Nurmi and J. Koolwaaij. Identifying meaningful locations. In *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*, pp. 1–8. 2006.

[24] M. Raento, A. Oulasvirta, et al. Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59, April 2005. ISSN 1536-1268.

[25] F. Reynolds. Camera phones: A snapshot of research and applications. 7(2):16–19, April–June 2008.

[26] B. Schilit, N. Adams, et al. Context-aware computing applications, 1994.

[27] A. Schmidt, K. A. Aidoo, et al. Advanced interaction in context. *Lecture Notes in Computer Science*, 1707:89–??, 1999.

[28] —. ilearn on the iphone: Real-time human activity classification on commodity mobile phones, 2008.

[29] J. Shen. Machine learning for activity recognition, 2004.

[30] D. Siewiorek, A. Smailagic, et al. Sensay: a context-aware mobile phone. In *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pp. 248–249. 2003.

[31] K. Song and Y. Wang. Remote activity monitoring of the elderly using a two-axis accelerometer. In *CACS Automatic Control Conference*. 2005.

[32] T. Stiefmeier, D. Roggen, et al. Wearable activity tracking in car manufacturing. 7(2):42–50, April–June 2008.

[33] M. Tentori and J. Favela. Activity-aware computing for healthcare. 7(2):51–57, April–June 2008.

[34] Y. Wang, J. Lin, et al. A framework of energy efficient mobile sensing for automatic user state recognition. In *MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 179–192. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-566-6.

[35] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and. Techniques with Java Implementations*, chap. 3, pp. 72–75. Morgan Kaufmann, 2000.