Institut für Visualisierung und Interaktive Systeme
Abteilung Intelligente Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis Nr. 3010

Recognition of motion patterns based on mobile device sensor data

Juan José Sanz Martín

Course of Study:	Infotech
Examiner:	Prof. Dr. Gunther Heidemann
Supervisor:	Dipl.-Inf. Julia Möhrmann
Commenced:	16^{th} February 2010
Completed:	16^{th} August 2010
CR-Classification:	I.2.6, I.4.8, I.5.4, F.2.2

Abstract

English

Recognition of motion patterns based on mobile device sensor data

The new generation of mobile phones contains a series of sensors for measuring acceleration, orientation and positions. Such sensor data have been widely used in the field of wearable computing to recognize postures and activities of persons. However, the recognition of activities based on the sensor data provided by one mobile device is still in its early stage.

The scope of this thesis is the development of an Android-based pattern recognition system for the classification of activities like walking, running, climbing stairs and descending stairs. The system uses three dimensional information from sensors for acceleration, orientation and magnetic field. Before the data is processed, the acceleration is normalized using the information provided by the orientation and magnetic field sensors. This normalization transforms the acceleration coordinate system to a new one with the z-axis pointing to the center of the earth and the y-axis to the north magnetic pole.

To recognize activities, motion and pose, a decision tree was used as classifier. The tree is trained with input objects (vectors) together with the desired output associated with that input. As input vectors, several features from the data acquired are extracted such as average, variance and range values from each of the three axes (x,y,z) in time domain. There is also another useful feature *the energy* calculated from the spectrum or frequency domain vector.

The results obtained were calculated for windows sizes of 1, 2 and 3 seconds of the normalized acceleration. As a first conclusion, window size of two seconds is the compromise solution since it has the best balanced recognition accuracy. The recognition rates, using two seconds as window size, are over 70% for activities such as running or walking and near 60% for climbing and descending stairs.

Español

Reconocimiento de patrones de movimiento a partir de informacion de sensores de dispositivos móviles

Las nuevas generaciones de teléfonos móviles contienen una serie de sensores para la medición de aceleración, orientación y posicionamiento. Esta información ha sido ampliamente utilizada en el campo de la computación portable (wearable computer) para reconocer posturas o actividades desarrolladas por personas. Sin embargo, el reconocimiento de actividades basadas en datos provenientes de sensores usando teléfonos móviles esta aun en una fase temprana.

Con el objetivo de desarrollar un sistema de reconocimiento de movimientos mediante los sensores incorporados en teléfonos móviles con el sistema operativo Android de Google, se implementó un sistema para el reconocimiento de actividades como andar, correr, subir escaleras y bajar escaleras. El sistema usa información tridimensional de los sensores de aceleración, orientación y campo magnético. Antes que los datos sean procesados, la aceleración se normaliza con la información proporcionada por los sensores de orientación y de campo magnético. Esta normalización transforma el sistema de coordenadas de la aceleración a un nuevo sistema en el que el eje z apunta al centro de la tierra mientras que el eje y apunta al polo norte magnético.

Para detectar actividades, movimiento y posturas, se ha utilizado un clasificador, exactamente un árbol de decisión. El árbol es entrenado con vectores de entrada junto con la salida asociada a esa entrada. Como vectores de entrada, se extraen varias características de los datos obtenidos como el valor medio, varianza y rango de cada uno de los tres ejes (x,y,z) para los vectores del dominio temporal. Otra característica especialmente útil es *la energía* calculada a partir del espectro o dominio frecuencial.

Los resultados obtenidos fueron hallados para tamaños de ventana de 1, 2 y 3 segundos. La primera conclusión a la que se ha llegado es que el tamaño de ventana de 2 segundos es la solución de compromiso ya que se obtienen unos resultados más balanceados. Las tasas de reconocimiento para el tamaño de ventana de 2 segundos superan el 70% en actividades como andar o correr, mientras que para subir o bajar escaleras rondan el 60%.

Contents

1. Introduction **9**
- 1.1. Related Work . . . 10
- 1.2. Project Scope . . . 12
- 1.3. Organization . . . 12

2. System Requirements **13**
- 2.1. Functional requirements . . . 13
  - 2.1.1. Data acquisition . . . 13
  - 2.1.2. Data Processing . . . 14
  - 2.1.3. Modes of Operation . . . 14
- 2.2. Non-functional Requirements . . . 15
  - 2.2.1. Performance . . . 15
- 2.3. Operating Environment . . . 16
  - 2.3.1. Sensors . . . 16
  - 2.3.2. Software Interfaces . . . 20
  - 2.3.3. Developing Software and Language . . . 20

3. System Design **21**
- 3.1. Overview . . . 21
- 3.2. Packages . . . 22
  - 3.2.1. Package: User Interface . . . 24
  - 3.2.2. Package: Functionality . . . 25
  - 3.2.3. Package: Sensors . . . 26
  - 3.2.4. Package: Features . . . 26
  - 3.2.5. Package: DataProcessing . . . 26
- 3.3. Data Acquisition . . . 27
  - 3.3.1. Environment conditions . . . 28
  - 3.3.2. Limitations . . . 28
  - 3.3.3. Data acquired . . . 28

4. Data Processing **31**
- 4.1. Normalization . . . 31
- 4.2. Re-sampling . . . 32
  - 4.2.1. Interpolation . . . 32
- 4.3. Time Domain Transformations . . . 34
- 4.4. Features . . . 34

5. Experimental Results **37**
5.1. Datasets 37
5.2. Recognition statistics 42
5.3. Discussions of results 44

6. Conclusions and Suggestions for Future Research **45**
6.1. Conclusions 45
6.2. Suggestions for Future Research 45

A. Glossary **47**

Bibliography **49**

List of Figures **54**

List of Tables **54**

1. Introduction

Motion recognition systems are computer software applications that try to determine human motion and posture with the aid of a wide range of hardware technologies such as cameras or sensors.

To cite a few of the innumerable applications of motion recognition systems, it can be mentioned biometrics, virtual reality, robotic control, virtual exercise, medical diagnosis and treatment, sports training, tracking and gaming systems.

In the beginning, mobile phones only incorporated microphones as useful hardware suitable for use as a noise sensor (e.g. WideNoise is an application to monitoring the noise levels [1]). Later, cameras and phones converged, adding another sensor that has been used for tasks such as reading QR Codes or two-dimensional bar codes (see Figure 1.1), object recognition for computer vision systems (e.g. computer vision system for the detection and identification of urban objects for tourist information services [2]) and even light sensors.

Figure 1.1.: QR Code suitable for being read by a mobile device equipped with camera.

When built-in GPS devices appeared on mobile phones, a wide range of new applications came out such as navigation systems, photo geolocalization, etc. Since then other kinds of sensors have been introduced, like temperature, proximity, ambient light (to extend battery life, adjusting the brightness of the screen to the environment), acceleration, orientation or

compass and magnetic field. For example, the augmented reality systems use the GPS device combined with a compass or orientation sensor to enhance the perception of reality (e.g. Augmented GeoTravel [3]) and accelerometer sensors are currently used to facilitate usability (automated rotation of the screen) and gaming.

In addition, with the increasing memory incorporated in new generations of mobile devices it is possible to perform, for example, image recognition tasks (e.g. Bar code recognition - Figure 1.1).

This project emerges from the fusion of two facts, previous successful works with sensors for recognition systems (see Section 1.1) and more and more complex sensors inside mobile devices. The main differences between this project and previous works are the use of a limited number of sensors, restricted by the built-in hardware equipped in the mobile device used, and the position of the sensors, since how the mobile phone is placed affects the data acquired from sensors. A placement can be specified but sensors will not be attached to a part of the body as in previous works.

1.1. Related Work

Considerable research has already been done about computer recognition systems for person motion or pose. There are works that try to approach this problem using recognition systems with cameras, wearable sensors and so on. Also, different pattern recognition solutions have been used in these systems ranging from simple classifiers to complex artificial neural networks (ANN).

Recognition systems based on cameras

About camera recognition systems, some works use a single camera providing input to a learning method, other works use several cameras and there are works that use an image-based rendering system to construct a 3-D model in order to recognize human motion.

Ben-Arie, Wang, Pandit and Rajaram [4] developed a method for the recognition of human actions and activities using videos recorded by only one camera. This method consists of representing activities by a set of pose and velocity vectors for hands, legs and torso. The results demonstrated that the method is robust in conditions of varying view angle in the range of 30 degrees. In other articles, Clarkson [5] and [6] analyze a system with a wearable camera and a microphone to distinguish situations (at work, in a subway or in a grocery store) and coarse events (in a conversation or walking down a busy street) .

To avoid the problem of viewing the subjects from a single viewpoint, the work done by the department of Computer Science and Engineering (University of Minnesota) [7] has come to the conclusion that cameras-based motion recognition systems have good results only when several cameras are placed in such a way that the images are taken orthogonally to the motion path.

Another even more sophisticated method was developed by Gavrila and Davis [8]. They use multiple synchronized cameras from multiple views to reconstruct the 3-D body pose at each time instant and then recognize the motion based on 3-D features.

All these related works search for a common objective that is to be able to have a machine performing the same recognition functions that humans do. This includes motion recognition too. Lindblad and Kinser [9] have researched this topic developing algorithms to isolate the important data for its analysis and processing using neuronal networks.

Recognition systems based on sensors

Other works seek the solution to computer recognition systems using one or several sensors placed in strategic fixed locations on the human body. This is the case in the work using accelerometers for ambulatory monitoring of subjects in the home environment for long-term periods [10]. The system uses only a single waist-mounted triaxial accelerometer that is used to feed a real-time classification system for human movement recognition. With this device, the system is able to distinguish between periods of activity and rest, the postural orientation of the wearer and detects events such as walking and falls. Then it can provide an estimation of metabolic energy expenditure.

In contrast, Bao [11] and the work done at the University of New South Wales in Sydney [12] employ several sensors distributed on different parts of the body. For the first one, five biaxial accelerometers were placed on the thigh, ankle, arm, wrist and hip, and from each of the sensors the mean, energy frequency-domain entropy and correlation of acceleration data were calculated on a sliding window of 6,71 seconds. Decision table, nearest neighbor, decision tree, and Naive Bayesian classifiers were tested with this features and the decision tree classifier showed the best recognition performance with an overall accuracy rate of 89% for activities such as walking, watching TV, and vacuuming. The work leads to the conclusion that specific training for each person can lead to higher recognition rates given sufficiently large training sets.

Looking for comfort in recognition systems using sensors, some works have been studied the possibility of placing sensors distributed in clothings. There are two different approaches for the wearable sensors problem, one with carefully positioned accelerometers and another using clothing saturated with small embedded sensors. When the quantity of sensors is increased, their individual information quality is decreased. Another problem is to find the optimal sensor location that is specially hard to discover beforehand in cloths. Works that investigate in this field are, for example, [13], [14], [15], [16] and [17].

Another example of sensor utilization is in the field of robotics, in which, more and more sensors have been used as a source of information. Especially in Japan, a lot of successful works have been done in the field of recognition systems for robotics, e.g. emotion recognition systems [18], gestures tracking [19] or human faces recognition [20].

1.2. Project Scope

The scope of the project is the development of an application for the Google Android mobile platform. This application will be used to recognize different human activities.

The way to achieve this goal is by using a device equipped with different kinds of sensors frequently used in the new generation of mobile devices, for example accelerometers and magnetometers, as well as other kinds of sensors such as GPS.

The tasks the application should perform are the following:

- Data acquisition for each of the activities to recognize in order to make up the training data set.
- Process the acquired data, i.e. transforming the original data sets into different domains such as frequency.
- Extracting features and building a vector with them that can be used to train, for example, a decision tree.
- Recognizing the activity of a person.

1.3. Organization

During this project, two applications have been developed for a phone with the Google's Android operating system:

- *Motion Acquisition* is a small application for Android phones that is used for acquiring data from sensors.
- *Motion Recognition* is the application that includes the data acquisition mode from the previous one, as well as, machine learning functions, in order to train a classifier that is used for recognizing activities.

This thesis is structured in chapters as it is shown below:

In the next chapter, **Chapter 2 – System Requirements**, functional requirements for the applications developed are described. Furthermore, the operating environment is specified.

Then, **Chapter 3 – System Design** contains the system design. This chapter includes diagrams (like application layer and package diagram), package and class descriptions. In this chapter is also explained how data has been acquired, as well as, specific limitations and conditions of the acquisition.

Chapter 4 – Data Processing contains how the data is processed and how the different features are extracted from this data.

Finally, in the last chapter, **Chapter 5 – Experimental Results**, there are the experimental results and conclusions.

2. System Requirements

This chapter contains the functional requirements of the applications *Motion Acquisition* and *Motion Recognition*.

2.1. Functional requirements

The system implements features for acquiring data, processing this data, training a classifier with the data processed and finally for recognizing acquired data using a classifier previously trained.

The activities to recognize are the following:

- Walk
- Run
- Climbing stairs
- Descending stairs

2.1.1. Data acquisition

Data acquisition was the first milestone, the importance of this step lies in the need of large datasets for the proper work of the next functionalities. For that reason, at the beginning, a small application (*Motion Acquisition*) was developed to achieve this primary goal early in the project. Later, this function was also included in the full application (*Motion Recognition*).

The information acquired for this function is acceleration, orientation and magnetic field data, retrieved from the device's sensors. During the acquisition of the data, the three data vectors are combined in order to obtain the acceleration normalized in reference to the orientation.

This normalization of the acceleration means that orientation and magnetic field data are used to retrieve acceleration data that is independent of changes in the orientation of the device. In this new coordinate system, one axis is always pointing to the center of the Earth and another to the north magnetic pole.

Furthermore, all the acquired data is also labeled to identify which activity was performed during the acquisition. (For more information about the data acquired see Chapter 3.3 – Data Acquisition)

In order to facilitate the utilization of data acquired, the data for an activity can be saved into a file. The information stored in the file is the data itself together with a context. The context includes an activity tag, the date and an optional description that the user of the application can write down.

2.1.2. Data Processing

The data processing is linked to either classifier training or activity recognition. In both the processing is the same. However, all this data processing depends on one variable parameter, the window size. This parameter determines how the vector is split into smaller parts (see Figure 2.1) that are used as input for the transformations and calculations in following phases. The split method uses overlapped windows with a fixed overlap of 50% which proved to be suitable in previous works.

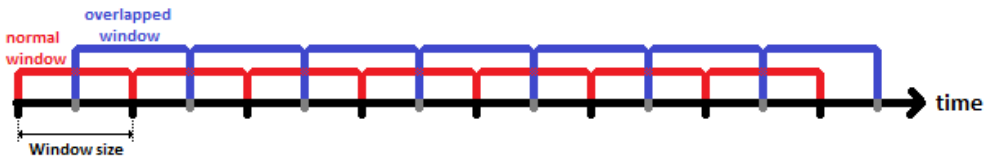

Figure 2.1.: Data processing using overlapping windows with overlapped window starting with an offset of 50% of the window size.

After the split phase, the time domain data is transformed into the frequency domain and then some features are extracted from both domains. The features extracted include average value, variance value and range for the time domain and energy for frequency domain. (See Chapter 4)

2.1.3. Modes of Operation

To recognize activities, a classifier, specifically a decision tree, is used to determine the correct output to an input, i.e. the activity that the user currently performs based on the sensor data. According to the functions, classifier training and activity recognition, there are mainly two modes of operation, learning mode (also training mode) and operation mode, which corresponds to the activity recognition.

Learning or Training Mode

In this mode of operation, the data previously acquired is processed and then the result is used as the input for the classifier to train it. The classifier, a decision tree, will be built in an iterative process that will add the new data to the previous entries.

In order to improve the classifier effectiveness, data sets should be large but also the quality of the data is very important. To avoid overfitting problems, it should be kept in mind that trees too big in size and complexity could be self-defeating. [21]

Operation Mode

In the operation mode, as well as in the learning mode, the data set needs to be processed, that is the features need to be extracted, and then the output will be used to feed the tree. The response of the tree should be the activity associated with the data set used as input. All the operations are performed without changing the tree structure, it means that, during the learning mode the tree is trained and in the operation mode the tree structure is not modified.

For statistics proposes, there are available two buttons that the user can use to count right and wrong recognitions. The recognition percentage can be observed in the screen.

2.2. Non-functional Requirements

2.2.1. Performance

Application's performance is an important parameter since an Android application will run on a mobile device with limited computing power and storage, and constrained battery life.

In the Developer's Guide that Google provides, there are the following basic rules for writing efficient code:

- Optimize judiciously
- Avoid creating objects
- Performance myths
- Prefer static over virtual
- Avoid internal getters/setters
- Use static final for constants
- Use enhanced for loop syntax

- Avoid enums where you only need ints
- Use package scope with inner classes
- Use floating-point judiciously

For more information consult the developer's guide [22].

In order to develop an efficient application, all the rules have been followed as far as possible and as appropriate, but also sometimes simplicity in the code has been prioritized over previous rules.

2.3. Operating Environment

The applications are developed to run on an Android device, to be precise the device is the *HTC Dream G1* that can be seen in the Figure 2.2.

Figure 2.2.: Android-powered Mobile phone HTC Dream G1, this is the device used during this project [23].

In the Table 2.1, all the technical specifications of the phone can be viewed [23].

2.3.1. Sensors

Android defines an interface for sensor hardware. The types of sensors currently supported by Android and integrated in the device used, *HTC Dream G1*, include the following. Note that all the sensors measures are in 3-axes, the definition of the coordinate system is also explained below. [24]

Processor	Qualcomm® MSM7201A™, 528 MHz
Operating System	Android™ 1.6
Memory	ROM: 256 MB RAM: 192 MB
Dimensions (LxWxT)	117.7 mm x 55.7 mm x 17.1 mm (4.60 in x 2.16 in x 0.62 in)
Weight	158 grams (5.60 ounces) with battery
Display	3.2-inch TFT-LCD flat touch-sensitive screen 320 x 480 (HVGA) resolution
Network	HSPA/WCDMA: Europe: 2100 MHz US: 1700/2100 MHz 7.2 Mbps down-link (HSDPA) 2 Mbps up-link (HSUPA) speeds Quad-band GSM/GPRS/EDGE: 850/900/1800/1900 MHz
Device Control	Trackball with Enter button
Keyboard	Slide-out 5-row QWERTY keyboard
Connectivity	Bluetooth® 2.0 with Enhanced Data Rate Wi-Fi®: IEEE 802.11b/g HTC ExtUSB™ (11-pin mini-USB 2.0 and audio jack in one)
Camera	3.2 megapixel color camera with auto focus
Audio	Built-in microphone and speaker
Battery	Rechargeable Lithium-ion battery (1150 mAh) Talk time: Up to 350 minutes for WCDMA Up to 406 minutes for GSM Standby time: Up to 402 hours for WCDMA Up to 319 hours for GSM
Expansion Slot	microSD™ memory card (SD 2.0 compatible)
AC Adapter	Voltage range/frequency: 100 240V AC, 50/60 Hz DC output: 5V and 1A
Orientation and location features	Accelerometer Digital Compass GPS receiver

Table 2.1.: HTC Dream G1 technical specifications [23]

- **Accelerometer**, all values are in SI units (m/s^2). For acceleration values of $(0, 0, 0)$, the device is defined such in free fall, while acceleration value proximate to $1g = 9.8m/s^2$, in one the axes, means that this axis is pointing to the Earth's center.

  The X axis refers to the screen's horizontal axis (the small edge in portrait mode, the long edge in landscape mode) and points to the right. See Figure 2.3.

  The Y axis refers to the screen's vertical axis and points towards the top of the screen (the origin is in the lower-left corner). See Figure 2.3.

  The Z axis points toward the sky when the device is lying on its back on a table. See Figure 2.3.

- **Orientation**, the coordinate system uses the Pitch, Roll and Azimuth reference system. This definition of yaw, pitch and roll is different from the traditional definition used in aviation where the X axis is along the long side of the plane (tail to nose).

  Azimuth, rotation around the Z axis $(0 \leq azimuth < 360)$. $0 = North$, $90 = East$, $180 = South$, $270 = West$. See Figure 2.4.

  Pitch, rotation around X axis $(-180 \leq pitch \leq 180)$, with positive values when the z-axis moves toward the y-axis. See Figure 2.5.

  Roll, rotation around Y axis $(-90 \leq roll \leq 90)$, with positive values when the z-axis moves toward the x-axis. See Figure 2.6.

- **Magnetic Field**, all values are in micro-Tesla (μT) and measure the ambient magnetic field in the X, Y and -Z axis.

- **GPS**

Figure 2.3.: Acceleration axes for the HTC Dream G1.

The application needs to define a listener interface to get all value changes produced in accelerometer, magnetometer and so on. And it is also necessary to define the sampling rate and accuracy.

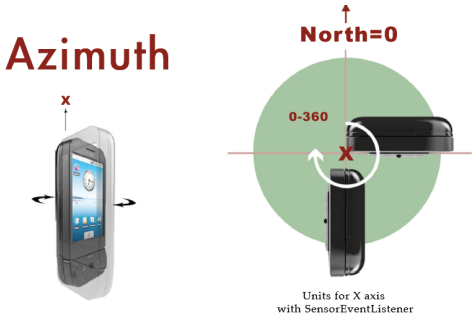

Figure 2.4.: Azimuth, rotation around the Z axis. [25]

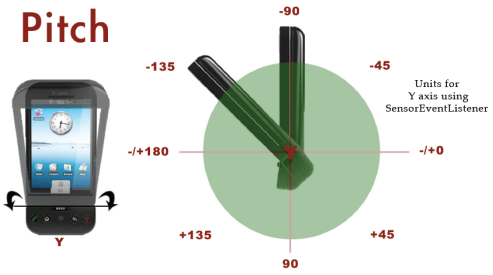

Figure 2.5.: Pitch, rotation around X axis. [25]

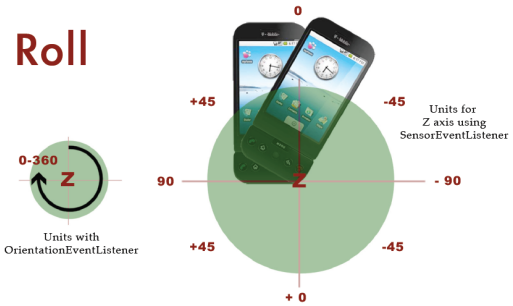

Figure 2.6.: Roll, rotation around Y axis. [25]

Android API defines four modes of sampling rate: [24]

- Fastest, to get sensor data as fast as possible.
  SENSOR_DELAY_FASTEST = 0
- Game, rate suitable for games.
  SENSOR_DELAY_GAME = 1
- Normal, rate (default) suitable for screen orientation changes.
  SENSOR_DELAY_NORMAL = 3

- User interface, rate suitable for the user interface.
  SENSOR_DELAY_UI = 2

When an application requests a specific sampling rate, it is really only a hint, or suggestion, to the sensor subsystem. There is no guarantee of a particular rate being available.

For accuracy there are also four options: [24]

- High, sensor reporting data with maximum accuracy.
- Low, sensor reporting data with low accuracy, calibration with the environment is needed.
- Medium, sensor reporting data with an average level of accuracy, calibration with the environment may improve the readings.
- Unreliable, the values returned by this sensor cannot be trusted, calibration is needed or the environment does not allow readings.

2.3.2. Software Interfaces

Android has different versions of the API that can be utilized depending on the device used or in case of the need to utilize new features only available in newer versions.

For this development, the version that fits better with the device used and gives a good interface for utilizing sensors is Android 1.5 Platform[1] (API Level: 3). Also this version is the default installed on the device used.

2.3.3. Developing Software and Language

The programming language used to develop applications on the Android platform is Java[2], furthermore Google provides the Android SDK[3] with tools and APIs.

The tools used in this project for the software developing are several,

- Eclipse IDE, Eclipse Classic version is recommended [27].
- ADT Plugin for Eclipse [28].
- Mercurial, for working with the repository and source control manager [29].
- Mercurial eclipse plugin [30].

[1] More information on `http://developer.android.com/intl/es/sdk/android-1.5.html`
[2] The Developer's Guide [24]
[3] Android SDK [26]

3. System Design

This chapter contains the system design. The design is described with the aid of diagrams (like application layer and package diagram), package and class descriptions.

For the design, a Model-View-Controller pattern has been followed. This software architectural pattern isolates the application logic for the user from input and representation or Graphical User Interface (GUI). The first advantage is that this architecture permits independent development, testing and maintenance for each of the parts [31].

3.1. Overview

The overall goal of the system is to provide a system for recognizing activities of a person carrying the android phone. For that purpose, the system designed, has been implemented using six layers, as it is shown in Figure 3.1.

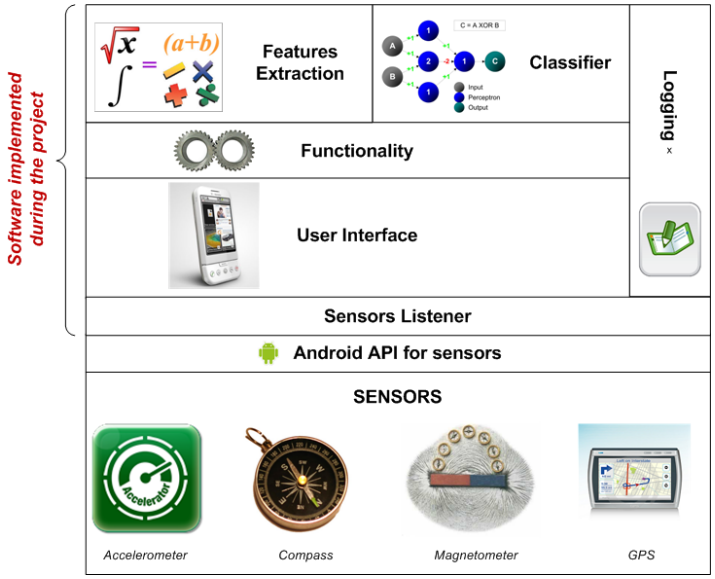

Figure 3.1.: Layer diagram of the application implemented divided by developed and previously existing layers.

In the above figure, there are two layers that already exist, *Sensors layer* and the *Android API layer*. These two layers represent the hardware related to sensors present in the phone and the API to access the data provided by Android, respectively.

Besides the GPS device, the hardware layer of the android phone utilized has the following sensors:

- AK8976A 3-axis Accelerometer
- AK8976A 3-axis Magnetic field
- AK8976A Orientation sensor
- AK8976A Temperature sensor

On the other hand, in Figure 3.1 it can be seen that the layers to implement are six:

1. Sensors listener layer, is the interface between the software developed and the API layer for sensors provided by Android (see Section 3.2.3).
2. User interface layer, for interaction between a human and an application (see Section 3.2.1).
3. Functionality layer, contains methods that create a new thread of execution to run common tasks such as acquiring data, processing the data and so on (see Section 3.2.2).
4. Features extraction layer, for mathematical or computational processing of the data which means features are obtained from input vectors. (see Section 3.2.4).
5. Classifier layer, generates a decision tree trained with a supervised learning method, and it is used to guess an unknown activity from the data acquired (see Section 3.2.5).
6. Logging layer, for having a centralized class for recording useful messages during the debugging and testing phase.

3.2. Packages

As it can be seen in the Figure 3.2, the package diagram contains one package for each of the layers to implement described in the above section.

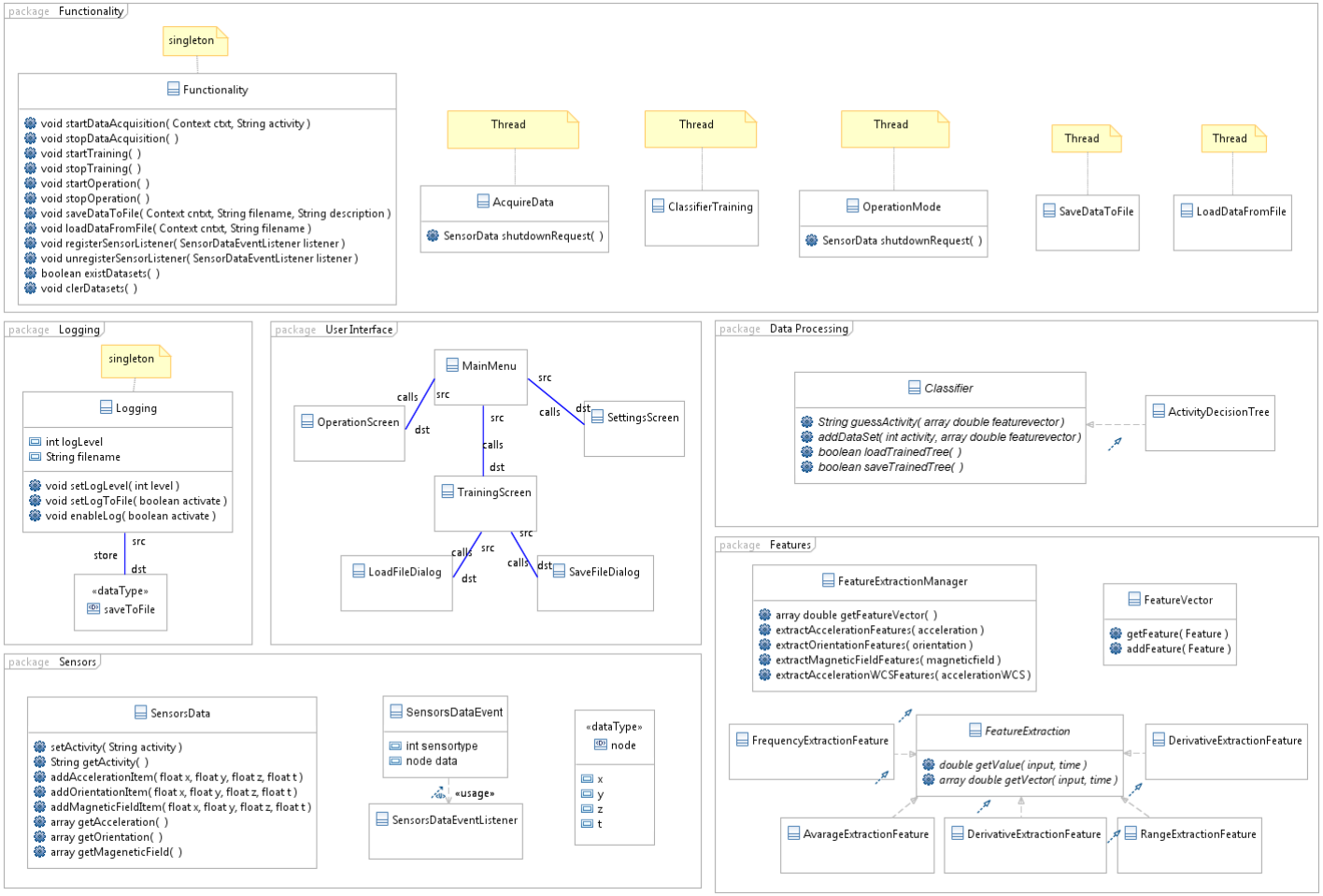

Figure 3.2.: Diagram with the application packages, classes with public methods and attributes.

3.2.1. Package: User Interface

This package realizes the user interfaces. User interfaces are provided for handling actions or showing results.

The screens in Figure 3.3 are the screenshots shown when the application is running. From the application shown in the screens, it is possible to request specific actions such as acquiring data, training the classifier or guessing an activity from data previously acquired.

(a) Operation Screen: Data acquisition for guess activities.

(b) Training Screen: The main function of this screen is to acquire data for training the classifier.

(c) Settings Screen: Parameters such as the path to store data and the window size can be set here.

Figure 3.3.: The three different views of the application.

In the Figure 3.3a, the operation screen can be seen. This view is used to start the data acquisition and process each window size of data acquired to guess which activity is being performed. For the proper work of this mode, a classifier should be previously trained with as much data as possible. A statistical analysis of recognition accuracy can be calculated using the existing buttons for that purpose.

Furthermore, the Figure 3.3b shows the training screen. Several functions can be performed from this screen, but the main functionality is to acquire data for training the decision tree classifier. An explanation of each button functionality is described in the following:

- Start Button: When this button is pressed, the data acquisition begins until the button is pressed again. The information acquired is stored in a part of the internal memory of the phone.
- Train Button: Uses the sensors data in the memory to train the decision tree classifier. The data in the memory could be acquired (with the start button) or loaded from a file that contains information previously acquired and saved (with the save data button).
- Test Button: This button has been added for testing functionalities. It uses the data from memory to guess which activity best matches.
- Train from dir Button: Uses all the files in a fixed path to train the classifier. The fixed path is */sdcard/media/TrainingSets/*.
- Test batch from dir Button: Loads all the files inside the path */sdcard/media/TestingSets/*, and uses each of them as the input to a trained decision tree to see if the output is the same as the ground truth.
- Save data Button: This button stores information from sensors previously acquired. The information is saved into a file than can be retrieved later. Furthermore, the corresponding activity label is also saved into the file.
- Load file Button: Loads sensors data from a file into memory. The file chosen should have been created using the previous button (Save data Button).

The Figure 3.3c shows the settings screen that is used to set some parameters such as the path to store data and the window size to process the data acquired.

3.2.2. Package: Functionality

Inside this package, classes that implement specific functions are included. Each of these functions are run in a new thread because they require heavy calculations and in this way they do not disturb the main thread that handles user and application events.

The class called *Functionality* acts as a centralized class to interact with the other packages and its purpose is to call each of the others classes that are run in a separate thread.

AcquireData is the class that manages the data acquisition, the data is stored in the memory of the phone. Then, the classes *SaveDataToFile* and *LoadDataFromFile* can operate with this memory data, the first one saving it into a file and the second loading data previously saved.

ClassifierTraining uses that current data in the memory to train the decision tree classifier with it. On the other hand, *ClassifierTesting* also uses the memory data but this time to test the decision tree output and compare it with the ground truth.

OperationMode is used for recognizing current activities. To do this, it acquires data and uses each window size of this data to look at the classifier for the appropriate output.

3.2.3. Package: Sensors

This package has two different kinds of classes, one for storing data and the other for generating custom events to notify the User Interface about any change in the sensors values.

The class *node* contains a custom data type that is used in the application for storing sensors data. It represent a point of the sensors data, each of these points have the information of the x, y and z axes and a time stamp. The time stamp denotes the time in seconds at which a point have been acquired. All the numbers are stored in the floating point format. Inside *SensorsData* the sensors information is stored using arrays formed with the data type mentioned before.

3.2.4. Package: Features

The class *FeatureVectorManager* is used as the centralized class that calls all the other classes. This centralized class uses the *FeatureVector* class to store the feature array (or feature vector) that will be used as input for training the decision tree (training set) or as input in a trained tree to guess which activity is associated to this input (guessing an activity).

This package contains also all the features extraction classes that extend the abstract class *FeatureExtraction*. The reason to implement classes extending an abstract class is for simplifying future extensions. The features extraction classes implemented are *Frequency Vector Extraction*, *Average Feature Extraction*, *Variance Feature Extraction*, *Range Feature Extraction* and *Energy Feature Extraction*.

Too see how the feature extraction process is done and which algorithms are used for extracting the features, see Chapter 4 – Data Processing.

3.2.5. Package: DataProcessing

This package includes the decision tree classification functionality.

There are basically two working behaviours for the decision tree, learning mode and operation mode. In the learning mode, the decision tree is trained with several inputs knowing the proper output (see Figure 3.4). Otherwise in operational mode, a trained tree should be able to select the correct output to a set of input values (see Figure 3.5).

An external library has been used to develop the decision tree, exactly jaDTi library [32]. This library has been chosen among others for being the lighter, computational speaking, and considering computer processing is a limiting factor in a mobile device.

The library used, jaDTi, has the following characteristics [32]:

- Read/Write from a database that is used to build a decision tree. The content of this database is called the learning set.

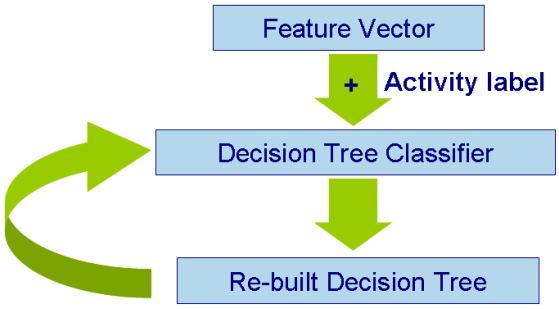

Figure 3.4.: Training Mode: The decision tree classifier is trained using an input and knowing the proper output.

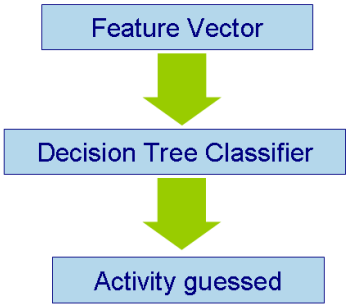

Figure 3.5.: Operation Mode: A trained tree returns the proper output for a set of input values.

- Build a decision tree using a learning set. A special attribute, the goal attribute, needs to be defined in order to guess its value.
- Draw the decision tree into a diagram.
- Guess the goal attribute of an item using the decision tree.

3.3. Data Acquisition

In this section, it will be explained how the data was acquired, as well as, the specific limitations and environment conditions of this project, as result of using built-in mobile phone sensors.

3.3.1. Environment conditions

3.3.2. Limitations

Basically there are two big differences between this data acquisition and the related work (Section 1.1).

1. The number of sensors used.
2. The position of the sensors since they are attached to the mobile phone device.

Other recognition systems use several sensors placed at fixed positions on the body. The handicaps of a recognition system using mobile phone devices are the limited number of sensors and finding a good position to place the device because the sensors are embedded in the device.

In order to define the position, the device will always be placed in one of the front pockets of the pants with the camera pointing to the front of the carrier. All the data have been acquired with the phone in a fixed position and variations in that position probably lead to bad recognition rates.

3.3.3. Data acquired

The following list of activities shows the activities for which training data has been acquired.

- Walking
- Running
- Climbing stairs
- Descending stairs

For the data acquisition, two different persons were carrying the phone in the front pocket of the pants. The data acquired has been obtained in different locations (e.g. different kinds of stairs and so on) to cover as many typical movements for each activity as possible.

In the Table 5.1, an approximate time can be seen of data acquired for each activity.

Activity	**Total time** (approximation)
Walking	30 minutes 45 seconds
Running	10 minutes 30 seconds
Climbing Stairs	16 minutes
Descending Stairs	16 minutes 15 seconds

Table 3.1.: Data sets acquired, total time is an approximation.

The process followed during the data acquisition, begins with selecting an activity in the application. To start the data acquisition the button start should be pressed and then data will be recorded until the stop button is pressed again. When the start button is pressed the application waits 5 seconds to start the data recording because the first seconds may contain not useful information and the carrier should place the phone in the defined position. For the same reason when the stop button is pressed, the last 3 seconds of the data acquired are discarded. After the acquisition process, the acquired data can be used or stored in a file.

4. Data Processing

Data processing normally involves seven phases: recording, analyzing, sorting, summarizing, calculating, disseminating and storing data. In this chapter, techniques of how the data is processed will be explained. The focus will be on the algorithms applied for feature extraction.

The data recorded needs to be transformed in order to obtain useful values that can be used in a classifier, the array or vector with these values is called *Feature Vector*. For that reason, the recorded data is transformed during a data transformation that can be summarized in the following steps:

1. Normalization: during the data acquisition the acceleration vectors are normalized using position and orientation information.
2. Re-sampling: data acquired must be re-sampled to obtain uniform time domain vector.
3. Domain transformations: from the time domain other vectors are extracted like frequency domain.
4. Features extraction: features will be extracted for each one the existing domains.

4.1. Normalization

During the data acquisition, information from three sensors is recorded giving data about acceleration, orientation and magnetic field. The acceleration data recorded is calculated in reference to a local coordinate system, and the other two sensors give information about the position of the phone on the earth, as well as the current orientation of the device. Therefore, in order to get even more useful information a normalization of the acceleration is done using the information provided by the other sensors to transform the local coordinate system to a new one in reference to the Earth (see Figure 4.1).

The acceleration normalization is performed at the same time as the data from the sensors is being recording. These normalized vectors are the input for the next steps that will be explained in the following sections.

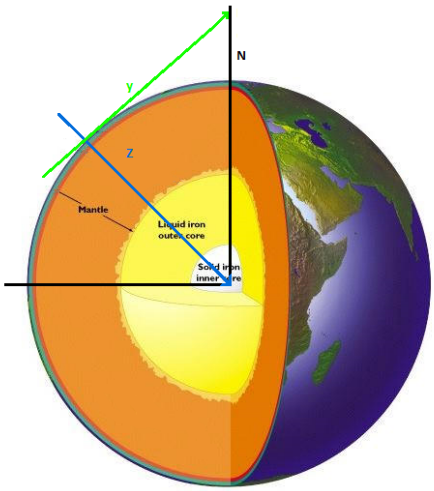

Figure 4.1.: Acceleration normalization, coordinate system transformation. The new coordinate system has the axis in reference to the Earth.

4.2. Re-sampling

The input data received is not guaranteed to be uniform in time, i.e. equidistant, for that reason before processing it, the data must be re-sampled. Interpolation techniques are used for re-sampling the data.

4.2.1. Interpolation

There are basically two different ways to approach this problem.

Linear Interpolation

Linear interpolation is the easier interpolation method. Consists in calculating the value of one point in the interval (x_0, x_1) knowing the two points in the edges (x_0, y_0) and (x_1, y_1) (see Figure 4.2). All the values (x, y) in the interval can be extracted from the following equation: [33]

$$y = y_0 + (x - x_0) \cdot (\frac{y_1 - y_0}{x_1 - x_0})$$

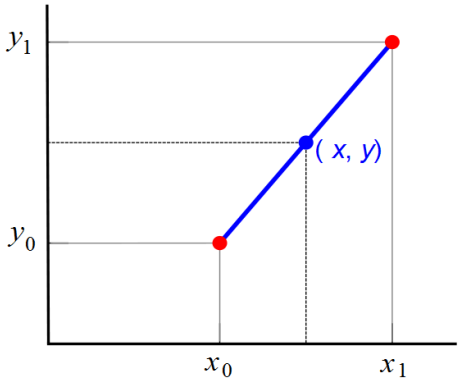

Figure 4.2.: Illustration of linear interpolation.

Polynomial Interpolation

Polynomial interpolation is the interpolation using a polynomial, i.e. given a data set, there is a polynomial which goes exactly through these points (see Figure 4.3). To find the polynomial, the following system has to be solved: [34]

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$
$$p(x_i) = y_i \qquad \text{for all } i \in \{0, 1, \ldots, n\}$$

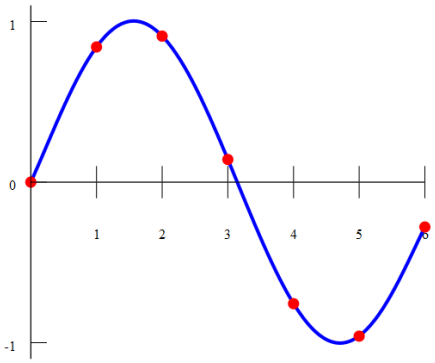

Figure 4.3.: Illustration of polynomial interpolation.

Algorithm used

The algorithm used to re-sampling data was linear interpolation. Basically, the reasons to choose the linear interpolation are the simplicity of the algorithm, and because of that, it is easier to process in a limited computational device as the one used.

4.3. Time Domain Transformations

All the data captured has been obtained in a time domain, but for detecting some signal characteristics other domains resolve the problem efficiently. A good example is energy calculation in frequency domain.

Frequency

Frequency domain or spectrum is calculated using a Fourier transformation. Time domain can be recovered from the spectrum using the inverse Fourier transformation. The theory says that all signals can be expressed as a sum of sinusoids, and the frequency domain decomposes a signal into its sine and cosine component.

The Fast Fourier Transform (FFT) is normally used in signal processing for efficient computation of the Fourier transform (FT) of finite length and uniformly-spaced signals.

Below is the formula used to calculate the FFT:

$$X(k) = \sum_{n=0}^{(N-1)} x(n) \cdot \mathrm{e}^{\frac{-j\pi kn}{N}}, \qquad \mathrm{o} \leq k \leq (N-1)$$

The spectrum is normally used for easier analyzing of repetitions in a signal but also it is commonly used for calculations like the energy of a signal.

4.4. Features

Since using the whole vectors is not computationally efficient for differentiating signals, some features are extracted from the domains and they will be used for the signal classification.

Average, range and variance are the features extracted from the time domain vectors. On the other hand, only energy is extracted from the frequency domain.

Average Value

Average value measures the arithmetic mean of an array.

Below is the formula used to calculate the average value:

$$\overline{x} = \frac{1}{n} \sum_{k=1}^{n} x_i$$

Variance Value

The variance value is used for distinguishing between probability distributions. The formula used is the following:

$$\sigma^2 = \frac{1}{n}\sum_{k=1}^{n}(x_i - \overline{x})^2$$

Range Value

The range is the difference between the largest and smallest values, in other words, the difference between the global or absolute maximum value and the global or absolute minimum value.

$$range = \max(x) - \min(x)$$

Energy

The energy value is calculated as the normalized sum of square frequency components.

$$Energy(freq) = \frac{1}{n}\sum_{k=1}^{n}(freq_i)^2$$

$$freq_i = \overbrace{\underbrace{x}_{\text{real}} + \underbrace{iy}_{\text{imaginary}}}^{\text{complex number}}$$

5. Experimental Results

5.1. Datasets

In this section some datasets from each activity have been chosen to be plotted. In general, graphics form different activities cannot be distinguished with the naked eye but some details can be commented. All the figures in this chapter have been plotted with Matlab.

Walk

In the Figure 5.1, it can be seen a graphical plot of the original walking data recorded from sensors. The figure contains three-axes data for acceleration, orientation and magnetic field in a local coordinate system. On the other hand, the Figure 5.2 represents the normalization of the acceleration vectors using the orientation and magnetic field to transform it to a new coordinate system referred to the Earth.

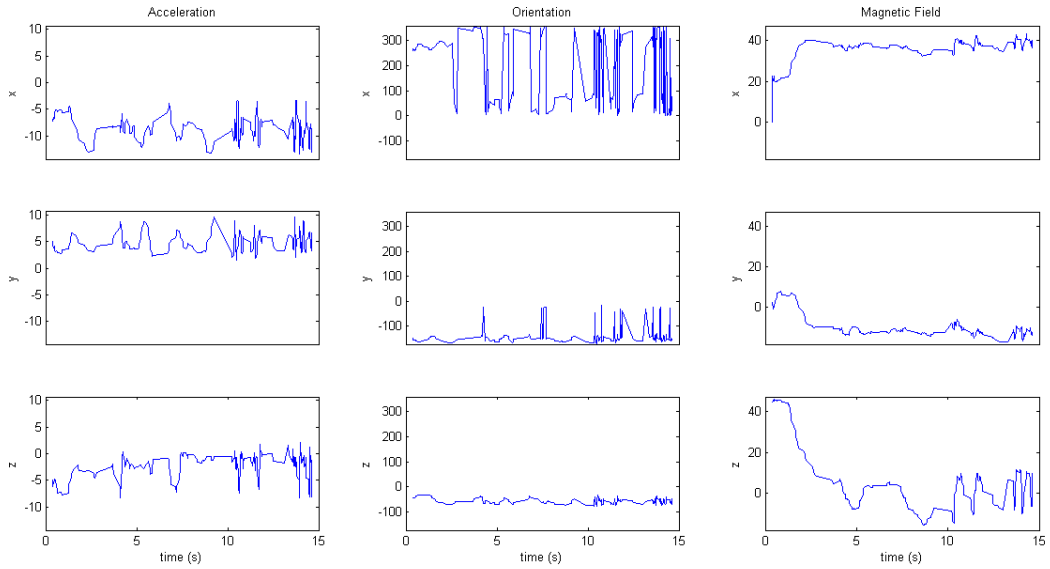

Figure 5.1.: Time domain acceleration, orientation and magnetic field data for the activity walking.

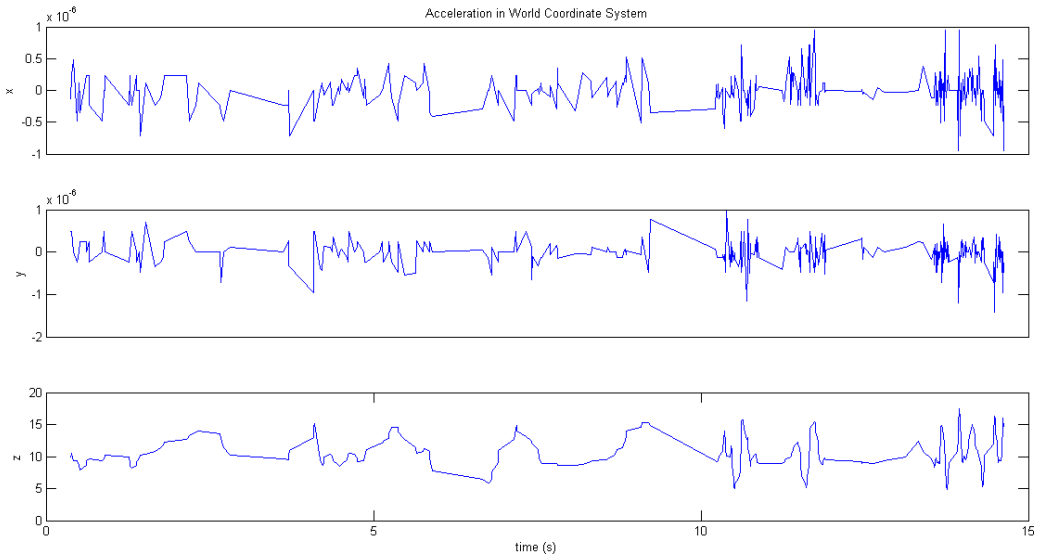

Figure 5.2.: Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity walking.

Comparing the walking and other activities graphics, it can be seen that the difference between maximum and minimum value of the acceleration vectors is normally lower than in the other activities.

Run

The Figure 5.3 and Figure 5.4 contain the original data recorded from the activity run in the first figure and the acceleration normalization in the second.

Great variations in the acceleration maximum and minimum values in short times are featured of running graphics but these features are similar to descending stairs and climbing stairs so initially it could be not easy to see with a naked eye.

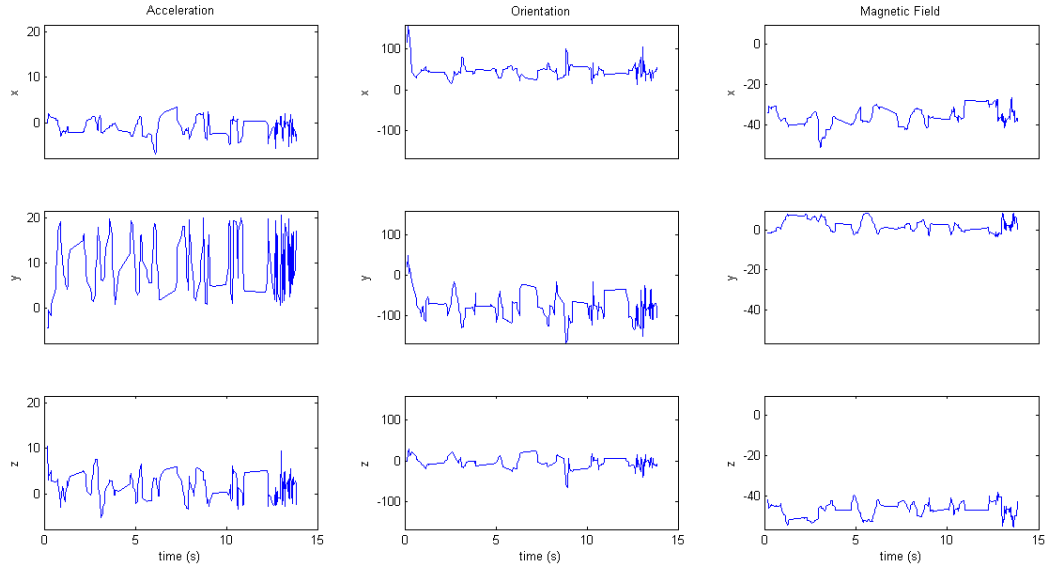

Figure 5.3.: Time domain acceleration, orientation and magnetic field data for the activity running.

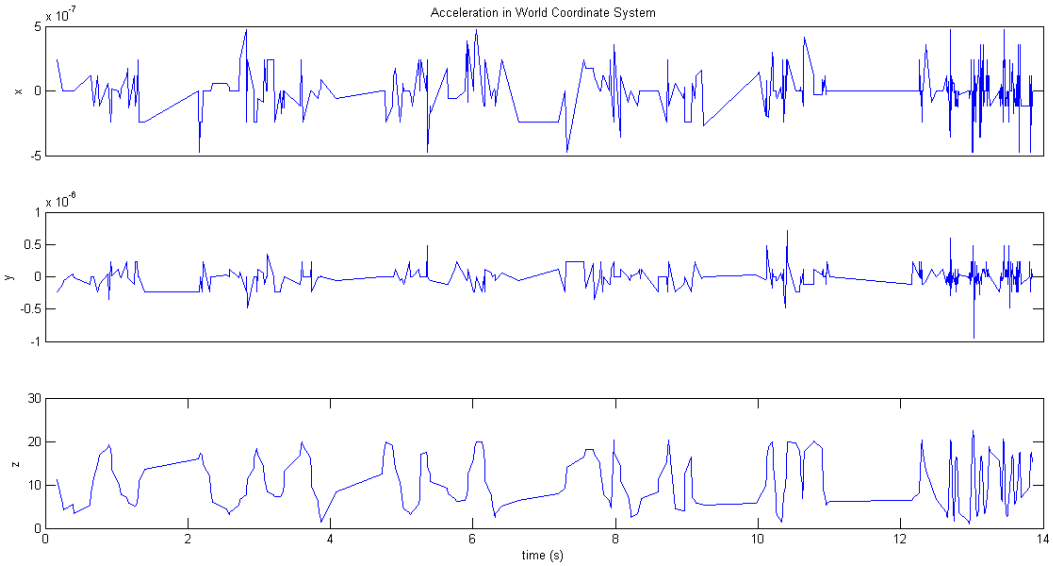

Figure 5.4.: Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity running.

Climbing stairs

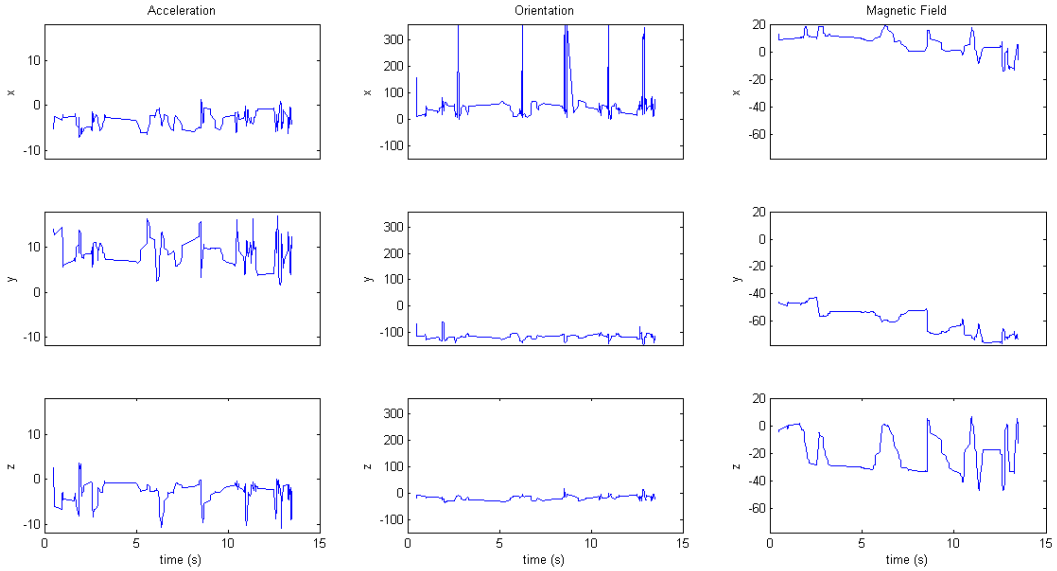

Figure 5.5.: Time domain acceleration, orientation and magnetic field data for the activity climbing stairs.

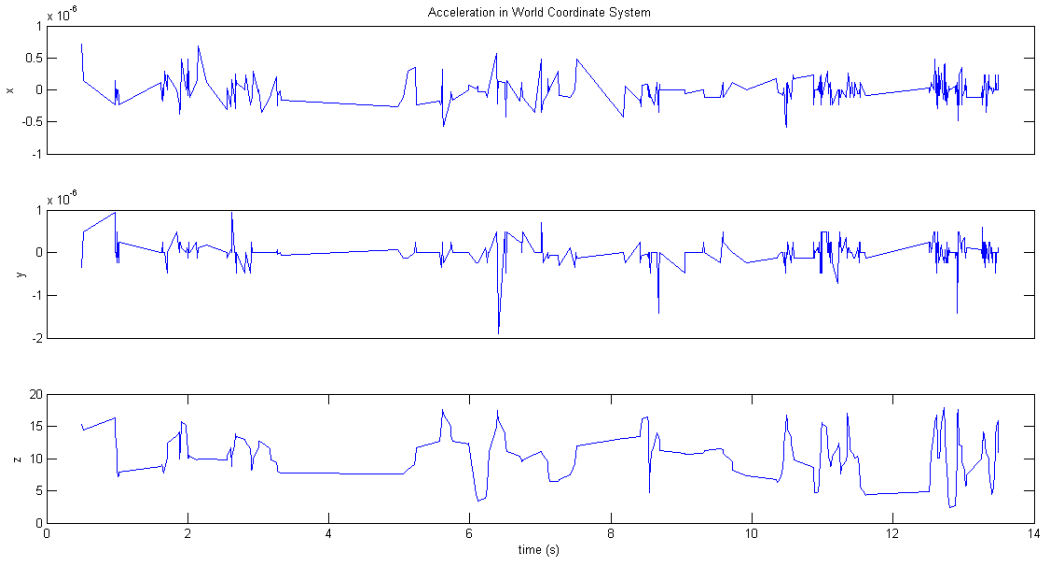

Figure 5.6.: Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity climbing stairs.

The Figure 5.5 and Figure 5.6 contain the original data recorded from the activity climbing stairs and the acceleration normalization respectively.

No significant differences can be found comparing the climbing stairs and descending stairs graphics.

Descending stairs

In the Figure 5.7 the original data recorded for the activity descending stairs can be seen. Moreover, the Figure 5.8 contains a representation of the normalized acceleration.

As it has been said before no significant differences can be found comparing the climbing stairs and descending stairs graphics.

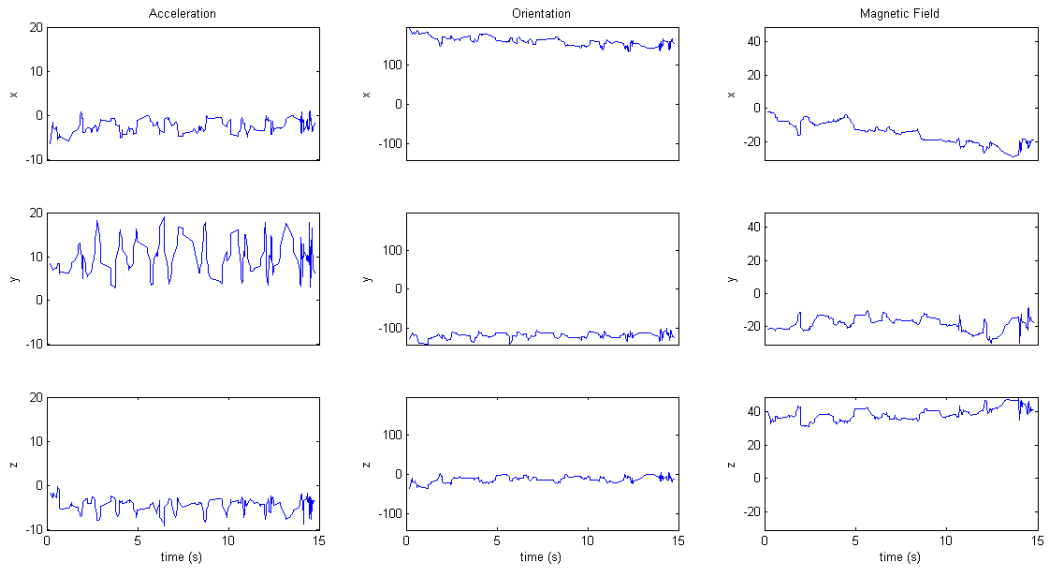

Figure 5.7.: Time domain acceleration, orientation and magnetic field data for the activity descending stairs.

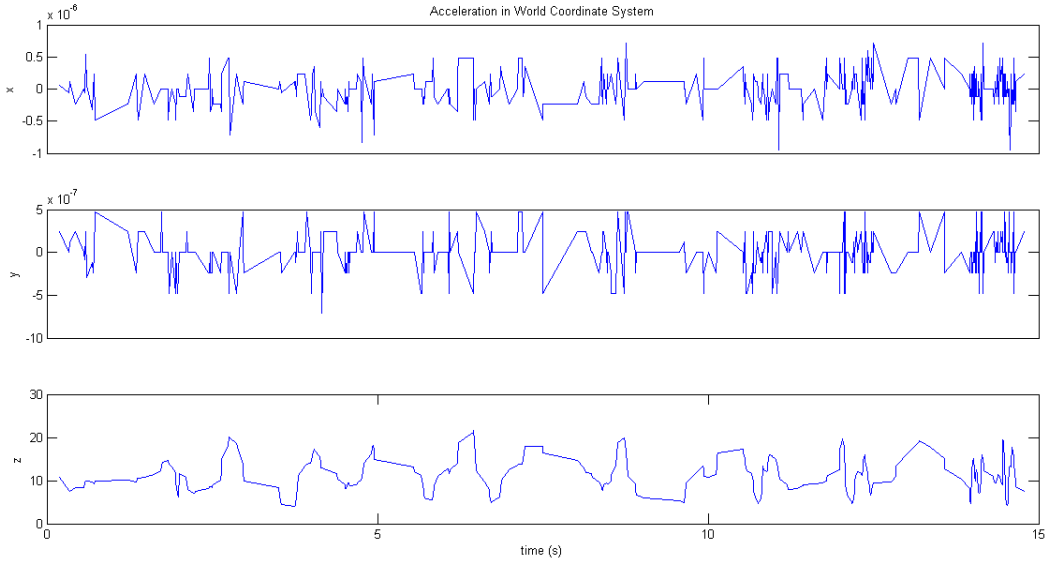

Figure 5.8.: Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity descending stairs.

5.2. Recognition statistics

The whole data set captured was used to obtain the statistics and the decision tree was trained using the 40% of the data sets. The statics were calculated setting window size of 1, 2 and 3 seconds with variable sample rate.

Later, new results were obtained using a window size of 2 second and a constant sample rate of 64 samples per second ($64 = 2^6$).

Activity	Training data sets	Testing data sets
Walking	738 seconds	1845 seconds
Running	252 seconds	630 seconds
Climbing Stairs	384 seconds	960 seconds
Descending Stairs	390 seconds	975 seconds

Table 5.1.: Data sets (in seconds) used for training the classifier (Training data sets) and for testing (Testing data sets).

Window Size: 1 second with variable sample rate

Ground truth / Activity	Walk	Run	Climbing Stairs	Descending Stairs
Walk	**67.4391%**	8.1697%	13.1579%	11.2333%
Run	17.7114%	**61.791%**	9.7512%	10.7463%
Climbing Stairs	26.9106%	8.7805%	**52.1951%**	12.1138%
Descending Stairs	24.8945%	8.6076%	12.0675%	**54.4304%**

Table 5.2.: Recognition accuracy statistics for window size of 1 second and variable sample rate.

Window Size: 2 seconds with variable sample rate

Ground truth / Activity	Walk	Run	Climbing Stairs	Descending Stairs
Walk	**61.3734%**	10.7296%	15.0215%	12.8755%
Run	16.9323%	**55.9761%**	13.9442%	13.1474%
Climbing Stairs	25.1445%	4.3353%	**58.0925%**	12.4277%
Descending Stairs	23.8372%	6.9767%	11.6279%	**57.5581%**

Table 5.3.: Recognition accuracy statistics for window size of 2 seconds and variable sample rate.

Window Size: 3 seconds with variable sample rate

Ground truth / Activity	Walk	Run	Climbing Stairs	Descending Stairs
Walk	**64.8069%**	7.8326%	15.0215%	12.3391%
Run	15.6923%	**59.6923%**	15.0769%	9.5385%
Climbing Stairs	28.1116%	6.0086%	**54.0773%**	11.8026%
Descending Stairs	20.8426%	6.2084%	17.2949%	**55.6541%**

Table 5.4.: Recognition accuracy statistics for window size of 3 seconds and variable sample rate.

Constant sample rate and window size of 2 seconds

Ground truth / Activity	Walk	Run	Climbing Stairs	Descending Stairs
Walk	**77.5484%**	2.5806%	10.3226%	9.5484%
Run	8.0679%	**81.9533%**	6.7941%	3.1847%
Climbing Stairs	22.3097%	4.7244%	**58.2677%**	14.6982%
Descending Stairs	20.3966%	6.7989%	13.881%	**58.9235%**

Table 5.5.: Recognition accuracy statistics for window size of 2 seconds and constant sample rate of 64 samples per second.

5.3. Discussions of results

Window size of 2 seconds has the best recognition accuracy for activities such as climbing stairs and descending stairs. On the other hand, for window size of 1 and 3 seconds walk and run activities have better recognition performance.

Then, the compromise solution would be using a window size of 2 seconds since it has the best average recognition accuracy for all the activities, best balanced recognition accuracy.

On the other hand, using a constant sample rate, activities like walking and running have a recognition rate over 70%, whereas climbing stairs and descending stairs have a recognition rate near 60%.

However, the recognition rates are far from good. This is due to several facts:

- Only simple features were used which describe aspects for each window size of the sensor data. Temporal aspects and more sophisticated correlation and frequency features were not investigated in this work due to lack of time.
- The chosen window sizes might not be appropriate for our task. Other successful works use larger window sizes of several seconds achieving better results.
- The sensor data provided by the mobile phone is prone to a lot of errors. Even though a sample rate and accuracy were set, there is not guarantee it will be fixed so it cannot be relied on. Furthermore, sometimes the sensors do not provide any information which leads to a series of zero-values. All this affects to the quality of the data sets acquired and as result to the recognition rates.
- Only two persons recorded motion data. And in previous works, it has been proved that employing several persons, the data sets obtained are more useful and diverse. This is because each person has different ways of performing an activity. Also increasing the data sets probably will lead to better results.
- The data pre-processing before extract features could not be enough, lineal interpolation was employed and depending the original sampling rate this technique could induce noise.

6. Conclusions and Suggestions for Future Research

6.1. Conclusions

In this thesis, it has been dealt with the issues of developing a motion recognition system.

First, an application was developed for gathering data from acceleration, orientation and magnetic field sensors of a phone, and with this application two persons were collecting data sets for the following activities: walking, running, climbing and descending stairs.

The data from the three sensors were combined to obtain acceleration vectors normalized using the orientation and magnetic field data. Theses normalized vectors were then splitted using overlapped windows of two seconds with a fixed overlap of 50% and re-sampled using interpolation techniques because the sample rate was not fixed.

At that point, each window was transformed into the frequency domain and from both domains some features were extracted. The features extracted are average, variance and range for the time domain and energy from the frequency domain.

Then, with this features a classifier were trained using the 40% of the acquired data, a decision tree classifier was chosen for its simplicity and effectivity in related works.

At the end, the classifier were tested with the acquired data obtaining recognition rates over 70% for activities like walking and running and near 60% for climbing and descending stairs.

These results were not too bad, however the recognition rates expected were around 80%. This is due to several facts such as the quality of the data obtained, the chosen window size and the extraction of only simple features.

6.2. Suggestions for Future Research

Regarding the work developed in this thesis, several research lines can be suggested for future work to improve the system results.

Changing the recognition system and/or the learning method the recognition rates should increase. In this project, a classifier has been used for recognizing activities. The learning method chosen for training the classifier was supervised learning. Supervised learning

means that during the training phase, the system is trained using input values and their known output. One suggestion of future research could be using neural networks that could lead to better recognition rates even though neural networks will introduce more complexity in the system. Other possibility is to used other learning methods like unsupervised learning that consists in learning from input patterns with no output values specified, it is done determining how the data are organized but probably the recognition rates will be worst than with supervised learning.

Another possibility to improve the results is adding advanced features extraction. These advanced features could be, for example, correlation between axes or integration of the acceleration to obtain the velocity. Also autocorrelation of the Fourier transform could be used for finding repeating patterns in a signal such as determining the presence of a periodic signal.

Also, data processing could be improved using polynomial interpolation instead of lineal interpolation or using filters before the features extraction that could reduce the noise in the sensors data.

In order to acquire better data from sensors and then enhance the results. Some control in the data accuracy and rate from sensors could be added to discard moments in which the data acquired quality could not be guaranteed.

A. Glossary

Throughout the document, some acronyms and conventions are used. The list of common terms and their acronyms is presented as follows:

ANN Artificial Neural Network.

API Application Programming Interface.

IDE Integrated Development Environment.

Decision Tree Classifier is a kind of classifier in which decision nodes are defined.

FFT Fast Fourier transform.

GUI Graphical User Interface.

IFFT Inverse Fast Fourier transform.

MVC Model–View–Controller.

SDK Software Development Kit.

UML Unified Modeling Language.

WCS World Coordinate System, coordinate system in which z-axis is always pointing to the Earth's center and y-axis to the north magnetic pole.

Bibliography

[1] T. D. L. A. R. O. Davide Casali, Aaron Brancotti, "Widenoise application that samples decibel noise levels." `http://www.widetag.com/widenoise/`. (Zitiert auf Seite 9)

[2] L. Paletta, G. Fritz, C. Seifert, P. Luley, and A. Aimer, "A mobile vision service for multimedia tourist applications in urban environments," in *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pp. 566 –572, 17-20 2006. (Zitiert auf Seite 9)

[3] AugmentedWorks, "Augmented geotravel." `http://www.augmentedworks.com/`. (Zitiert auf Seite 10)

[4] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, "Human activity recognition using multidimensional indexing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1091–1104, 2002. (Zitiert auf Seite 10)

[5] B. Clarkson, K. Mase, and A. Pentl, "Recognizing user's context from wearable sensors: Baseline system," tech. rep., Massachusetts Institute of Technology Media Laboratory, 2000. (Zitiert auf Seite 10)

[6] B. Clarkson and A. Pentl, "Predicting daily behavior via wearable sensors," tech. rep., 107 Author Index Penev, Tsvetan, 1 Queisser, Marcel, 57 Reza Soleymani, Hamid, 73 Twellmeyer, Eva, 33 Vagts, Hauke-H, 2001. (Zitiert auf Seite 10)

[7] O. M. Robert Bodor, Bennett Jackson and N. Papanikolopoulos, "Image-based reconstruction for view-independent human motion recognition." `http://mha.cs.umn.edu/Papers/IBR_MotionRecognition.pdf`. (Zitiert auf Seite 10)

[8] D. M. Gavrila and L. S. Davis, "3-d model-based tracking of humans in action: a multi-view approach," in *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, (Washington, DC, USA), p. 73, IEEE Computer Society, 1996. (Zitiert auf Seite 11)

[9] T. Lindblad and J. M. Kinser, *Image Processing Using Pulse-Coupled Neural Networks*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. (Zitiert auf Seite 11)

[10] N. H. L. Merryn J. Mathie, Adelle C. F. Coster and B. G. Celler, "Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement," *Physiological Measurement*, vol. 25, 2004. (Zitiert auf Seite 11)

[11] S. S. Intille, L. Bao, and L. Bao, "Physical activity recognition from acceleration data under seminaturalistic conditions," tech. rep., Thesis, Massachusetts Institute of Technology, 2003. (Zitiert auf Seite 11)

[12] N. Wang, E. Ambikairajah, S. J. Redmond, B. G. Celler, and N. H. Lovell, "Classification of walking patterns on inclined surfaces from accelerometry data," in *DSP'09: Proceedings of the 16th international conference on Digital Signal Processing*, (Piscataway, NJ, USA), pp. 357–360, IEEE Press, 2009. (Zitiert auf Seite 11)

[13] K. V. Laerhoven and H.-W. Gellersen, "Spine versus porcupine: A study in distributed wearable activity recognition," *Wearable Computers, IEEE International Symposium*, vol. 0, pp. 142–149, 2004. (Zitiert auf Seite 11)

[14] S.-W. Lee and K. Mase, "Activity and location recognition using wearable sensors," *IEEE Pervasive Computing*, vol. 1, 2002. (Zitiert auf Seite 11)

[15] H. w. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artefacts," *Mob. Netw. Appl*, vol. 7, pp. 341–351, 2002. (Zitiert auf Seite 11)

[16] K. V. Laerhoven, A. Schmidt, and H. werner Gellersen, "Multi-sensor context aware clothing," in *in Proceedings of the Sixth International Symposium on Wearable Computers, 2002*, pp. 49–56, IEEE Press, 2002. (Zitiert auf Seite 11)

[17] K. V. Laerhoven and O. Cakmakci, "What shall we teach our pants?," 2000. (Zitiert auf Seite 11)

[18] J.-W. Hong, M.-J. Han, K.-T. Song, and F.-Y. Chang, "A fast learning algorithm for robotic emotion recognition," in *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*, pp. 25 –30, 20-23 2007. (Zitiert auf Seite 11)

[19] M. Gonzalo-Tasis, C. Laschi, J. Finat, and P. Dario, "Monocameral visual recognition of marcus hand postures for personal robotic assistants," in *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, vol. 2, pp. 1464 – 1467 vol.2, 2001. (Zitiert auf Seite 11)

[20] W. Huang, B. Lee, M. Rajapakse, and L. Li, "Face recognition by incremental learning for robotic interaction," in *Multimedia Signal Processing, 2002 IEEE Workshop on*, pp. 280 – 283, 9-11 2002. (Zitiert auf Seite 11)

[21] M. Sebban, R. Nock, J. H. Chauchat, and R. Rakotomalala, "Impact of learning set quality and size on decision tree performances," *Int. Journal of Computers, Systems and Signals*, vol. 1, pp. 85–105, 2000. (Zitiert auf Seite 15)

[22] Google, "Android developer's guide." `http://developer.android.com/guide/index.html`. (Zitiert auf Seite 16)

[23] HTC, "Htc mobil phones - t-mobile g1." `http://www.htc.com/us/products/t-mobile-g1`. (Zitiert auf den Seiten 16, 17, 53 und 54)

[24] Google, "Android reference." `http://developer.android.com/intl/es/reference/packages.html`. (Zitiert auf den Seiten 16, 19 und 20)

[25] Novoda, "Motion sensors and orientation in android." http://novoda.com/2009/05/02/motion-sensors-and-orientation-in-android/. (Zitiert auf den Seiten 19 und 53)

[26] Google, "Android sdk." http://developer.android.com/intl/es/sdk/index.html. (Zitiert auf Seite 20)

[27] E. Foundation, "Eclipse - downloads." http://www.eclipse.org/downloads/. (Zitiert auf Seite 20)

[28] Google, "Adt plugin for eclipse." http://developer.android.com/intl/es/sdk/eclipse-adt.html. (Zitiert auf Seite 20)

[29] M. community, "Mercurial distributed source control management tool." http://mercurial.selenic.com/. (Zitiert auf Seite 20)

[30] V. Z. Andersen), "Mercurial plugin for eclipse." http://www.vectrace.com/mercurialeclipse/. (Zitiert auf Seite 20)

[31] T. Reenskaug, "The model-view-controller (mvc)." http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf. (Zitiert auf Seite 21)

[32] J.-M. François, "jadti - decision trees: a java implementation." http://www.run.montefiore.ulg.ac.be/~francois/software/jaDTi/. (Zitiert auf Seite 26)

[33] K. encyclopedia, "Linear interpolation." http://www.knowledgerush.com/kr/encyclopedia/Linear_interpolation/. (Zitiert auf Seite 32)

[34] K. encyclopedia, "Polynomial interpolation." http://www.knowledgerush.com/kr/encyclopedia/Polynomial_interpolation/. (Zitiert auf Seite 33)

All links were last followed on Sunday 30^{th} July 2010.

List of Figures

1.1. QR Code suitable for being read by a mobile device equipped with camera. . . 9

2.1. Data processing using overlapping windows with overlapped window starting with an offset of 50% of the window size. 14
2.2. Android-powered Mobile phone HTC Dream G1, this is the device used during this project [23]. 16
2.3. Acceleration axes for the HTC Dream G1. 18
2.4. Azimuth, rotation around the Z axis. [25] 19
2.5. Pitch, rotation around X axis. [25] 19
2.6. Roll, rotation around Y axis. [25] 19

3.1. Layer diagram of the application implemented divided by developed and previously existing layers. 21
3.2. Diagram with the application packages, classes with public methods and attributes. 23
3.3. The three different views of the application. 24
3.4. Training Mode: The decision tree classifier is trained using an input and knowing the proper output. 27
3.5. Operation Mode: A trained tree returns the proper output for a set of input values. 27

4.1. Acceleration normalization, coordinate system transformation. The new coordinate system has the axis in reference to the Earth. 32
4.2. Illustration of linear interpolation. 33
4.3. Illustration of polynomial interpolation. 33

5.1. Time domain acceleration, orientation and magnetic field data for the activity walking. 37
5.2. Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity walking. 38
5.3. Time domain acceleration, orientation and magnetic field data for the activity running. 39
5.4. Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity running. 39
5.5. Time domain acceleration, orientation and magnetic field data for the activity climbing stairs. 40
5.6. Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity climbing stairs. 40

5.7. Time domain acceleration, orientation and magnetic field data for the activity descending stairs. 41
5.8. Time domain acceleration normalized using a new coordinate system referred to the Earth for the activity descending stairs. 42

List of Tables

2.1. HTC Dream G1 technical specifications [23] 17

3.1. Data sets acquired, total time is an approximation. 28

5.1. Data sets (in seconds) used for training the classifier (Training data sets) and for testing (Testing data sets). 42
5.2. Recognition accuracy statistics for window size of 1 second and variable sample rate. 43
5.3. Recognition accuracy statistics for window size of 2 seconds and variable sample rate. 43
5.4. Recognition accuracy statistics for window size of 3 seconds and variable sample rate. 43
5.5. Recognition accuracy statistics for window size of 2 seconds and constant sample rate of 64 samples per second. 44

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Juan José Sanz Martín)