TÍTOL DEL TFC: Design and Implementation of a Tourist Application for Mobile Devices

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTHOR: Fausto Gil Moreno

DIRECTOR: Francesc Tarrés Ruiz

DATE: June, 19th 2012
Overview

The aim of this project is to build a travel application for smartphones. This application will create a touristic video made with pictures from Panoramio. The name of the application is VARMAPP (Video and Route Application).

For that purpose, we have created VARMAPP Server that manages communications between the smartphone and the Google Maps and Panoramio servers.

In the application, the user selects the departure and arrival cities and the server looks for the best route in Google Maps. After that, a map with the desired route is displayed.

Once VARMAPP server receives information about GPS coordinates and selected route, a request to Panoramio server is sent demanding for photos around the route in order to create a video containing locations of interest or touristic places near this route.

Finally, a musical video containing those images is displayed in the smartphone. The video can be seen in false 3D using Red Cyan glasses.

This project is divided in 3 main sections.

In the first section, we will study how OS Android works and the design of the graphic interfaces in the application.

In the second section, we will discuss how devices communicate with each other in the network and how they obtain information from other devices. We will see how the smartphone connects with VARMAPP Server and how VARMAPP Server manages the information link between Google Maps and Panoramio servers. This communication process is the responsible of obtaining significant data that will be used in the creation of the final video.

Finally, in the third section, we will study how to process and apply random transition effects to the photographs obtained from Panoramio and will also examine the program used to create the video, the Avidemux video editor.

The aim of this project is to develop an application for tourist travelling and photos amateurs for Android smartphone users, so they can enjoy a new audiovisual experience.
This project has marked a before and an after in my life.

I would like to dedicate this project to my family, specially my sister and my wife who have always encouraged me throughout in my academic career and make possible for me to stand where I am today. Thanks to them I am a better person.

I would like to thank Renat Rius for supporting me from the very beginning.

I thank my supervisor, Professor Francesc Tarrés, for many insightful conversations during the development of the ideas in this project, and for helpful comments on the text.

I greatly appreciate the help from Israel Delgado, without his advice this project wouldn’t have been possible to realize.

This project is dedicated to them with all my love.
INDEX

INTRODUCTION.................................................................................................................. 6

CHAPTER 1. MOBILE APPLICATION DEVELOPMENT IN ANDROID OS ..... 8
1.1 Introduction.................................................................................................................. 8
1.2 OS Android.................................................................................................................. 8
   1.2.1 History................................................................................................................ 8
   1.2.2 Architecture......................................................................................................... 10
1.3 Android developed environment .............................................................................. 12
   1.3.1 JDK (Java SE Development Kit)......................................................................... 12
   1.3.2 SDK (Software Development Kit) ........................................................................ 12
   1.3.3 ADT (Android Development Tools) .................................................................... 12
   1.3.4 Eclipse ................................................................................................................ 13
1.4 The Video and Route Application ............................................................................. 14
   1.4.1 Introduction ......................................................................................................... 14
   1.4.2 Graphic interfaces............................................................................................... 14
1.5 Summary .................................................................................................................... 16

CHAPTER 2. SYSTEM ARCHITECTURE AND MODULE COMMUNICATIONS
............................................................................................................................................... 17
2.1 Introduction.................................................................................................................. 17
2.2 Checking internet connection.................................................................................... 18
2.3 General communications diagram............................................................................ 19
   2.3.1 Smartphone - Google Maps Server Link ............................................................ 20
   2.3.2 Smartphone - VARMAPP Server Link ............................................................... 22
   2.3.3 Creating searching areas .................................................................................... 23
   2.3.4 VARMAPP Server Link – Panoramio Server Link (I) ......................................... 25
   2.3.5 VARMAPP Server Link – Panoramio Server Link (II) ....................................... 28
   2.3.6 Smartphone - VARMAPP Server Link ............................................................... 30
2.4 Summary .................................................................................................................... 31

CHAPTER 3. VIDEO EFFECTS PROCESSING AND CODING IN THE
VARAMPP SERVER.............................................................................................................. 32
3.1 Introduction.................................................................................................................. 32
3.2 Images processing ..................................................................................................... 32
   3.2.1 Resize.................................................................................................................. 32
   3.2.2 Effects................................................................................................................ 33
3.3 Video codec................................................................................................................. 41
   3.3.1 Script.................................................................................................................. 43
   3.3.2 Testings.............................................................................................................. 45
3.4 Summary ........................................................................................................... 48

CHAPTER 4. GENERAL CONCLUSIONS ....................................................... 49

4.1 Environmental impact ....................................................................................... 50

4.2 Future lines ....................................................................................................... 51

BIBLIOGRAPHY ..................................................................................................... 52
INTRODUCTION

Mobile devices have come a long way since the introduction of the smartphone some years ago. Smartphones have become essential for modern living, they allow us to check e-mails, surf the internet, take pictures and shoot video and many other things. There are thousands of different applications for smartphones and there is where the idea of this project comes from.

The aim of this project is to develop a travel application for Android smartphone users, so they can enjoy an audiovisual experience. In order to achieve that, we have created VARMAPP Server that processes communications between the smartphone and the Google Maps and Panoramio servers.

In the application interface, the user selects the departure and arrival cities and Google Maps creates a route that is shown in a map. Then, a video made with the most recent pictures from Panoramio is displayed. This video can be seen in false 3D using Red Cyan glasses. The name of this application is VARMAPP (Video and Route Application).

This application has been developed in Android because nowadays, OS Android has become the most widely used mobile operating system for mobile devices.

Developing this application requires several tasks, including learning the tools needed to create or edit videos and the study of the OS Android systems and servers. This Project is divided into four chapters where key issues will be discussed. All the chapters are interrelated in the sense that each chapter contains information necessary for the understanding of subsequent sections.

In the first chapter, we will study how OS Android works. Among others, we will analyze the architecture, the tools and the advantages of this operating system. We will also have a look at the graphic interfaces designed for this application.

In the second chapter, we will discuss how devices communicate with each other in the network and how they obtain information from other devices. We will see how the smartphone connects with VARMAPP Server and how VARMAPP Server connects with the Google Maps and Panoramic Servers. This communication process is the responsible of obtaining significant data that will be used in the creation of the final video.

Following, in the third chapter, we will study how to process and apply random transition effects to the photographs obtained from Panoramio.

The final result will be an amusing and entertaining video that is going to be different every time due to the random nature of the data mining process in the servers. We will also examine the program used to create the video, the Avidemux video editor.
Finally we have chapter four. The objective of this last chapter is to critically review the system we have developed. We will propose some future lines and analyze the environmental effects of this application.

As it has been said at the beginning of this chapter, there are thousands of smartphones applications then, why create a new one?

The answer is simple; there is no application at the moment that can create a video between two places selected by the user.

We think that this audiovisual experience may be interesting for many people. The idea is to provide the user with an automatic selection of photos based on the availability of these photos in a web server taking into account the route programmed by the same application.

In order to study the availability of the system, we have evaluated and implemented some of possible ideas for the application because too much time would be needed to implement all the possibilities.

Future work may include advertising of restaurants, commercial centers, etc. that can be found in the route.

Summarizing the aim of VARMAPP is to provide the users with a new application that allows them to choose a departure and an arrival city and watch an updated video between these two places.

As there are so many applications for smartphones it is difficult to build a find space for an application that is not already in the market. The purpose of this project is to offer something different and useful for users who are willing to try new applications.
CHAPTER 1. MOBILE APPLICATION DEVELOPMENT IN ANDROID OS

1.1 Introduction

The aim of this introductory chapter is to provide a brief description of the Android operative system. In this paper we present a general view about its history, how it is structured and the required tools to program Android. At the end of the chapter, we will analyze the graphic interfaces designed specifically for this project application.

1.2 OS Android

1.2.1 History

Android [1] is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It was developed by the Open Handset Alliance, led by Google, and other companies. Google purchased the initial developer of the software, Android Inc., in 2005. The unveiling of the Android distribution in 2007 was announced with the founding of the Open Handset Alliance, a consortium of 86 hardware, software and telecommunication companies devoted to advancing open standards for mobile devices. Google releases the Android code as open-source, under the Apache License. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android.

Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. Developers write primarily in a customized version of Java. Apps can be downloaded from third-party sites or through online stores such as Google Play (formerly Android Market) or the Amazon AppStore. In October 2011, there were more than 500,000 apps available for Android, and the estimated number of applications downloaded from the Android Market as of December 2011 exceeded 10 billion.

Android was listed as the best-selling smartphone platform worldwide in Q4 2010 by Canalys with over 300 million Android devices in use by February 2012. According to Google's Andy Rubin, as of December 2011, there were over 700,000 Android devices activated every day. By February 2012 this had accelerated to 850,000 per day and boosted the year-on-year growth rate of Android to over 250%.

 According to the International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker, smartphones powered by the Android and iOS mobile operating systems accounted for more than eight out of ten smartphones shipped in the first quarter of 2012 (1Q12). The mobile operating systems
Android and iOS held shares of 59.0% and 23.0% respectively of the 152.3 million smartphones shipped in 1Q12.

**Android Operating System Highlights**

Android finished the quarter as the overall leader among the mobile operating systems, accounting for more than half of all smartphone shipments. In addition, Android boasted the longest list of smartphone vendor partners. The chart 1.1 below shows the android market share versus other mobile operating systems [2].

**Chart 1.1.** Top six smartphone operation systems, shipments and market share

<table>
<thead>
<tr>
<th>Mobile Operating System</th>
<th>1Q12 Unit Shipments</th>
<th>1Q12 Market Share</th>
<th>1Q11 Unit Shipments</th>
<th>1Q11 Market Share</th>
<th>Year-over-Year Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>89.9</td>
<td>59.0%</td>
<td>36.7</td>
<td>36.1%</td>
<td>145.0%</td>
</tr>
<tr>
<td>iOS</td>
<td>35.1</td>
<td>23.0%</td>
<td>18.6</td>
<td>18.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td>Symbian</td>
<td>10.4</td>
<td>6.8%</td>
<td>26.4</td>
<td>25.0%</td>
<td>-50.6%</td>
</tr>
<tr>
<td>BlackBerry OS</td>
<td>9.7</td>
<td>6.4%</td>
<td>13.8</td>
<td>13.6%</td>
<td>-29.7%</td>
</tr>
<tr>
<td>Linux</td>
<td>3.5</td>
<td>2.3%</td>
<td>3.2</td>
<td>3.1%</td>
<td>9.4%</td>
</tr>
<tr>
<td>Windows Phone 7/Windows Mobile</td>
<td>3.3</td>
<td>2.2%</td>
<td>2.6</td>
<td>2.6%</td>
<td>25.9%</td>
</tr>
<tr>
<td>Other</td>
<td>0.4</td>
<td>0.3%</td>
<td>0.3</td>
<td>0.3%</td>
<td>33.3%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>152.3</strong></td>
<td><strong>100.0%</strong></td>
<td><strong>101.6</strong></td>
<td><strong>100.0%</strong></td>
<td><strong>49.9%</strong></td>
</tr>
</tbody>
</table>

The following figure shows an illustration of the chart 1.1.

**Fig. 1.1** Illustration of data from Chart 1.1.
1.2.2 Architecture

The following diagram (Fig. 1.2) shows the major components of the Android operating system architecture. It consists of four different layers: Applications, Application Framework, Libraries, Android Runtime and Linux Kernel [3].

![Fig. 1.2 O.S. Android Architecture](image)

**Application Framework**

Android ships with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using Java.

Underlying all applications is a set of services and systems, including:

- A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.
- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification manager that enables all applications to display custom alerts in the status bar
- An Activity manager that manages the lifecycle of applications and provides a common navigation back stack.
Libraries

Android includes a set of C/C++ libraries used by various components of the Android system.

- Surface Manager – manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
- Media Libraries – support playback and recording of many popular audio and video formats, as well as static image file.
- FreeType – bitmap and vector font rendering.
- SQLite – a powerful and lightweight relational database engine available to all applications.

Android Runtime

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.

Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included “dx” tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.
1.3 Android developed environment

As it has been said before, Android is a Linux-based operating system for mobile devices. In the next section, we’ll have a look at the software requirements for developing Android applications. Some of these requirements are noted below [4].

1.3.1 JDK (Java SE Development Kit)

The Java Development Kit (JDK) [5] is an Oracle Corporation product aimed at Java developers. Since the introduction of Java, it has been by far the most widely used Java Software Development Kit (SDK). Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. While less common on desktop computers, Java applets are sometimes used to provide improved and secure functions while browsing the World Wide Web.

1.3.2 SDK (Software Development Kit)

A software development kit (SDK or “devkit”) [6] is typically a set of software development tools that allows for the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform.

It may be something as simple as an application programming interface (API) in the form of some files to interface to a particular programming language or include sophisticated hardware to communicate with a certain embedded system. Common tools include debugging aids and other utilities often presented in an integrated development environment (IDE).

1.3.3 ADT (Android Development Tools)

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.

ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application.
Figure 1.3 shows the emulator Visual interface.

![Fig. 1.3 GUI emulator](image)

1.3.4 Eclipse

Eclipse [7] is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java.

The initial codebase originated from VisualAge, The Eclipse SDK (which includes the Java development tools) is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Characteristics

- Multi-platform (GNU/Linux, Solaris, Mac OSX, Windows)
- Supported by different architectures
- Extensible plug-in system
- SVN or subversion version control (with subclipse).

Syntax highlighting, auto complete, selected code block tabulator, several edition utilities that make the programmer’s job easier.

In Eclipse, most data is created using wizards. A wizard is an assistant that guides you step-by-step through a process, for example creating new resources or importing and exporting them.
Advantages

- It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Haskell, Perl, PHP, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy and Scheme
- Memory usage a bit lower than other developed environments
- Available versions for almost any SO (operating system) including Linux, Windows and MAC OS X.
- Multilanguage
- Uses little hard drive space.
- Portable
- Excellent labelling assistant.

In the next section, we will analyze the graphics interfaces created for this application.

1.4 The Video and Route Application

1.4.1 Introduction

In this section the main application developed in this project will be described in detail. The name of the application is VARMAPP (Video and Route Application).

At the moment, there are several applications for searching pictures, reading maps, and so on; the aim of VARMAPP is to go a step further creating a video between two places selected by the user. The video is created using images collected in the internet by a process of data mining popular photographic social networks such as Panoramio.

1.4.2 Graphic interfaces

VARMAPP is made up of two graphic interfaces. Those interfaces have been designed in an intuitive and friendly way providing an easy and comfortable navigation.

The following figure 1.4, shows the main interface of VARMAPP that will appear when we run the application. The interface contains a welcome message followed by a brief explanation of how the application works and two textboxes.
In one of them, we will enter the name of the departure city and in the other the name of the arrival city.

![Graphic interface main application menu](image1.png)

**Fig. 1.4** Graphic interface main application menu

After selecting the cities, the user accesses the second graphic interface. In this interface a small screen that shows the instruction/route between the selected cities is found together with a brief explanation for the user (at the right top), some buttons and a checkbox.

![Graphic interface secondary menu application](image2.png)

**Fig. 1.5** Graphic interface secondary menu application

After pressing the image, the second interface allows the user to see the route graphically (example route Barcelona - Tarragona). A brief explanation can be found in the right top of the screen. This route can be changed anytime by pressing the "New route" button.

Then, the application creates a new video with the cities previously selected by the user in the main interface of VARMAPP. This video is saved and played automatically. The user has the option to watch the video in false 3D by selecting the false 3D button.
In this chapter we have analyze Android and the graphic interfaces of VARMAPP, in the next chapter, we will study the communication between the different devices for doing possible this project.

1.5 Summary

Summarizing this chapter, we can conclude that:

- The reason why we have chosen OS Android for our project is because this operative system has spread worldwide and it’s one of the best-selling Smartphone platforms.

- Programming an Android application is not as easy as it could seem but, like other programming languages, can be learned through practice.

- VARMAPP graphic interfaces have been designed in an intuitive and friendly way providing an easy and comfortable navigation.
CHAPTER 2. SYSTEM ARCHITECTURE AND MODULE COMMUNICATIONS

2.1 Introduction

In this chapter we will have an overview of the different devices that are required for the optimum performance of the application. We will also examine how they communicate with each other.

This is one of the longest chapters of the project, as it is the basis for optimum application performance. Before we begin, let’s define some terms that are necessary for the understanding of chapter content.

API

An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. An API specification can take many forms, including an International Standard such as POSIX or vendor documentation such as the Microsoft Windows API, or the libraries of a programming language, e.g. Standard Template Library in C++ or Java API.

In the execution of the application, we have used API’s from Google Directions and Panoramio.

A request of the route is done with the Google Directions API. This API will provide us with the data that will be sent later to the Panoramio server. After this process, the server receives images from the route.

Panoramio API provides the VARMAPP server with the necessary images to generate a video that will be displayed in the smartphone.

JSON

JSON or JavaScript Object Notation [8] is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for many languages.

The JSON format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML.
JSON is used when making a request to the Google Maps Server. The information that receives the smartphone is a file with several parameters of the route. From all these data, the application is programmed to select only the latitude and longitude parameters as those are the ones needed for this application.

We have also used the JSON format when making a request from the VARMAPP to the Panoramio server. The information that receives the VARMAPP server is a file that contains several parameters about the images found in the desired route. Some of those parameters are: the images data, title, author, graphic file URL, etc. After filtering this file, we obtain the images addresses that will be used for the VARMAPP server in order to obtain the images.

**POI**

We can define Point Of Interest (POI) as the number of collection latitude + longitude received from the Google Maps server in JSON format when the VARMAPP server makes a request.

Depending on the route selected by the user, the JSON file that we receive will have more or less POI’s. The number of POI’s is directly connected with the selected route. Therefore, as longer is the route, as higher is the number of POI’s obtained.

### 2.2 Checking internet connection

The application has been created in a way that when we start VARMAPP the first thing that the application does is checking the internet connection. If there is no connection, VARMAPP generates a message informing the user that an internet connection is required for running the application.

For a correct functioning of the application it is very important to add permissions in `AndroidManifest.xml`, without them the application won’t work. The following permissions have been added in the `AndroidManifest.xml` of the application:

**Internet**

<uses-permission>“android.permissionINTERNET“

**Network state**

<uses-permission>“android.permission.ACCESS_NETWORK_STATE “
2.3 **General communications diagram**

This general diagram shows an overview of the communications between the different devices that are required for the optimum performance of the application and defines their functions.

![General communications diagram](image)

**Fig. 2.1 General communications diagram**

Before explaining the different parts that compose the general diagram of communications, we will define the devices that appear in the diagram and their functions.

**Smartphone**

The Smartphone is used to make requests to the Google Maps server. In the screen of the smartphone, the user will see the map of the selected route and after that, a video of the route will be displayed.

**Google Maps Server**

This is the server that provided us with the data relating to the POI's of the route selected by the user in the smartphone.
VARMAPP Server

This is the server programmed for the application. It is the responsible to connect with the Panoramio server to obtain the images and connect with the smartphone when sending the video. This server is programmed in C#.

Panoramio Server

In a first request, this server provides us with data relating to the images of the desired route. After the VARMAPP server has filtered these data, the Panoramio server makes a second request that will return us the images that can be found in the selected route.

It is essential to have in mind the relationship between the different components of the system in the general diagram if we want to understand how the application works. For a better description of the diagram, we will divide it in five LINKs that will be explained in the following LINK.

2.3.1 Smartphone - Google Maps Server Link

![Communications diagram Smartphone - Google Maps Server Link]

This LINK is the responsible for requesting the route to API Google Directions and after that extract longitude and latitude values.

Link Overview and Specifications

The smartphone sends a HTTP request to the Google Maps Server. Then, the server will send back a file in JSON format with the data of our query. The details of these protocols are given below.
Google Maps API requests

The Google Directions API is a service that calculates directions between locations using an HTTP request [9].

An example of request to the Google Directions API can be seen below:

A Directions API request takes the following form:


where output may be either of the following values:

- json (recommended) indicates output in JavaScript Object Notation.
- xml indicates output as XML.

In this case output = json

Request parameters

In this section, we will explain the parameters that compose the request made to the Google Maps server for the application.

VARMAPP request - Google Maps Server

http://maps.google.com/maps/api/directions/json?origin=+value+&destination=+valuee+&a void=highways&sensor=false

A list with some of the parameters and their possible values are enumerated below.

- **origin**: the address or textual latitude/longitude value from which you wish to calculate directions.

In this case, “+value+” -> name of the departure city entered in the textbox.
Design and Implementation of a Tourist Application for Mobile Devices

- **destination**: the address or textual latitude/longitude value from which you wish to calculate directions.

  In this case, "+valuee+" → name of the arrival city entered in the text box.

- **avoid**: indicates that the calculated route(s) should avoid the indicated features.

  Currently, this parameter supports the following two arguments:
  
  - **Tolls**: indicates that the calculated route should avoid toll roads/bridges.
  - **Highways**: indicates that the calculated route should avoid highways.

  In this case avoid = highways, the program will find a route off the highway.

- **sensor**: Indicates whether or not the directions request comes from a device with a location sensor. This value must be either true or false.

  In this case, sensor = false.

There are other parameters as mode, way points, alternatives, units and region that are not in the list above. These parameters have not been explained as they haven’t been used in this project.

After receiving the JSON, the next step is to extract the desired values. The process is explained in the following LINK.

### 2.3.2 Smartphone - VARMAPP Server Link

![Fig. 2.4 Communications diagram Smartphone - VARMAPP Server Link](image)

This LINK is the responsible for extracting latitude and longitude values. After the extraction, they are sent to VARMAPP server for later analysis.
Link Overview and Specifications

The smartphone sends longitude and latitude values to VARMAPP Server. There, they are classified into longitude and latitude strings so that VARMAPP Server Link – Panoramio Server Link (I) can use them to obtain the addresses of the images used in making the video.

![Smartphone sending longitude and latitude to VARMAPP Server](image)

**Fig. 2.5** Smartphone - VARMAPP Server

In the next section, we will analyze how the latitude and longitude values are used to create searching areas.

### 2.3.3 Creating searching areas

Once the data from the previous LINK is processed, we will obtain a set of points of interest along the desired route. How many POI's we have depends on the route chosen by the user.

The four values needed in the API Panoramio request are minx, miny, maxx and maxy (minimum longitude, latitude, maximum longitude and latitude). Therefore, it has been necessary to create an image SAR (Search Area Ratio) that once applied to the POI's gives us minx, miny, maxx and maxy values.

An example is explained below:

**Route:** Barcelona-Tarragona

VARMAPP Server obtains a set of POI's in this route, 35 are found in this case.

In this example, we have selected the first point of interest, POI 1.

**POI 1** → latitude: 41.387410º, longitude: 2.168560º
When applying \text{SAR}^* (\text{Search Area Ratio}) = 0.001, the obtained values are:

Minimum longitude (\text{minx}): \text{longitude} - \text{SAR} = 2.168560^\circ - 0.001 = 2.16756^\circ  
Minimum latitude (\text{miny}): \text{latitude} - \text{SAR} = 41.387410^\circ - 0.001 = 41.38641^\circ  
Maximum longitude (\text{maxx}): \text{longitude} + \text{SAR} = 2.168560^\circ + 0.001 = 2.16956^\circ  
Maximum latitude (\text{maxy}): \text{latitude} + \text{SAR} = 41.387410^\circ + 0.001 = 41.38841^\circ  

*By varying the coefficient value, the searching area is widened or decreased.*

The resulting area is shown in the figure below (Fig. 2.6)

![Image](image_url)

**Fig. 2.6** Searching area in Google Maps

As it can be seen in the figure, the result is approximately a 140 meters sides square.

For the measuring of the created area, we have used Maps Labs from Google that is a tool that allows you to measure distances in a map.

Default values have been used in \text{SAR}. In the future, it would be interesting to program a module in the software to execute an automatic POI’s processing that would select more or less depending on the route and would also select the searching areas. It has been impossible to implement this module as more time would be necessary to do it.

After obtaining the \text{minx}, \text{miny}, \text{maxx} and \text{maxy} values, the next step is to use this data to do a request to the API Panoramio.
2.3.4 VARMAPP Server Link – Panoramio Server Link (I)

![Communications diagram VARMAPP Server Link – Panoramio Server Link](image)

**Fig. 2.7** Communications diagram VARMAPP Server Link – Panoramio Server Link

**Link Overview and Specifications**

VARMAPP Server sends a request to Panoramio Server that will send back a file in JSON format with the data of the request.

![HTTP request diagram VARMAPP Server – Panoramio Server](image)

**Fig. 2.8** VARMAPP Server – Panoramio Server

**Panoramio**

Panoramio is a geolocation-oriented photo sharing website. Accepted photos uploaded to the site can be accessed as a layer in Google Earth and Google Maps, with new photos being added at the end of every month. The site’s goal is to allow Google Earth users to learn more about a given area by viewing the photos that other users have taken at that place.

Most of the images uploaded to the Panoramio website are from Europe with a number of uploads specially high in Spain.
In the heat map below, we can see the places with a higher number of images uploads [10].

![Heat Map](image-url)

**Fig. 2.9 Map**

**API requests**

API requests to Panoramio are restricted to 100,000 a day and the maximum number of images for request is 100. If any of those values are exceeded in a period of 24 hours the server could stop working temporarily [11].

This restriction doesn’t affect the design of the application because we are using a prototype and the number of request by day is not high. Therefore, it would be necessary to have this fact into account when developing this application in a professional environment.

API Panoramio uses JSON as the exchange format for images. Through this API a requesting URL is created and Panoramio sends back a JSON file with the images data, title, author, graphic file URL, etc.

The data from section 2.3.3 is used to do a request.

**Request parameters**

In this section, we will analyze the parameters that compose the request that the application makes to the Panoramio server.
The lines below shows an example of a type request to API Panoramio.

VARMAPP Server Request – Panoramio Server


- **set**
  - *public* (popular photos)
  - *full* (all photos)
  - *user ID number*

In this case, **set** = *public*.

When selecting set = public, the images will be automatically updated in Panoramio, therefore, the video will always contain the most recent images of every place. The *full* option haven’t been selected because in doing as a result we would obtain not updated videos.

- **from/to**

You can define the number of photos to be displayed using "from=X" and "to=Y", where Y-X is the number of photos included. The value 0 represents the latest photo uploaded to Panoramio. These images are automatically organized from more to less recent.

After several trials, we have decided that the ideal values for **from** and **to** are 0 and 5 respectively (from = 0, to = 5). These values are based on the results obtained in the trials. As the **from** and **to** values increase, the volume of the images contained in the video also increases and the result is that the video obtained is too big. When the **from** and **to** values decrease, we will obtain a small video if there are a few images in the POI’s because some images could be lost. **minx, miny, maxx, maxy** define the area to show photos from (minimum longitude, latitude, maximum longitude and latitude, respectively).

- **size**
  - original
  - medium (default value)
  - small
  - thumbnail
  - square
  - mini_square

In this case **size** = *medium*

The ideal size for the application is medium that correspond to 500x300 pixels images. The size of the images is approximate; the exact size depends on the size of the original image.
That is the size that is closer to the screen size of smartphones (800x480 pixels).

After the request, the next step is to continue with the images acquisition process.

### 2.3.5 VARMAPP Server Link – Panoramio Server Link (II)

![Diagram of VARMAPP Server Link – Panoramio Server Link](image_url)

**Fig. 2.9** Communications diagram VARMAPP Server Link – Panoramio Server Link (II)

This LINK is the responsible for requesting photographs to the Panoramio Server.

**Link Overview and Specifications**

The VARMAPP Server makes a request of the photographs that interest us to the Panoramio Server. Then, the Panoramio Server sends to VARMAPP Server the images that will be used later.

![Diagram of VARMAPP Server – Panoramio Server](image_url)

**Fig. 2.10** VARMAPP Server – Panoramio Server
At the end of this section, VARMAPP Server will have the images needed to create a video.

It is important to have in mind that Panoramio will send us 5 images for every POI of the selected route. As the Barcelona-Tarragona route includes 35 POI’s, we will receive five images for POI 1, five more for POI 2 and so on until we reach 35 POI. That means that at the end we will have a total of 175 images. These will be the images used in the creation of the resulting video.

In the next picture shows the images for POI 1 in the route Barcelona-Tarragona.

Fig. 2.11 Shows images for POI 1 in the route Barcelona – Tarragona
2.3.6 Smartphone - VARMAPP Server Link

This LINK is the responsible for processing the received photographs.

Link Overview and Specifications

Among other functions, VARMAPP Server classifies, resizes and applies transitions to the photographs. After that process is finished, it creates a video that is sent to the smartphone.

The creation of the video from photograph sequence requires the generation of some automatic effects and the compression of the video in an efficient coder. This process will be explained in detail in the next chapter.
2.4 Summary

Summarizing this chapter, we can conclude that:

- Also, it has been necessary to learn how to do request to the Google Maps and Panoramio servers.

- Establishing a communication infrastructure between the devices used for this project is extremely complicated and challenging. Several hours have been spent in the development of the infrastructure.

- The time of response of the different links used is quite short and, after several trials, no problems have been found in this field.
CHAPTER 3. VIDEO EFFECTS PROCESSING AND CODING IN THE VARAMPP SERVER

3.1 Introduction

In the previous chapter, we have analyzed how the system of communication works. This system consists of five sections, four of which have already been explained previously. The fifth section is the subject of this chapter, whose purpose will be to examine the processing of images in the VARAMPP server.

At the end of the chapter, we will explain how to coding the video that will be reproduced on the smartphone. The name of the program used for that purpose is Avidemux.

The processing of the images to produce the animations and effects in the video has been programmed from scratch. Although some of the effects have been inspired by other photo-viewer applications others have been specifically devised for our application.

3.2 Images processing

Once the VARMAPP server obtains the images from Panoramio a processing process starts. The result of this process is a video between two cities selected by the user.

This video is generated with the images obtained in the POI’s of the route.

In the following lines, we will explain the different processes that are applied to the images before generating the final video that will be displayed in the smartphone.

3.2.1 Resize

As it has been said before, current smartphones resolution is around 800x480 pixels. That is the reason why in the Panoramio request the size \( \rightarrow \) medium that correspond to 500x300 pixels approximately has been selected. As not all the images are the same size, the VARMAPP server resizes them to 800x480 pixels. So we will have a set of images with the same size to create a video.

In order to do the resizing of the photographs, a program has been created. This program, which is inside the VARMAPP server, is the responsible of resizing and listing the images.

As it has been said before, the most of the smartphones have a resolution of 800x480 pixels approximately but there are others with different resolution and therefore, the resize of the images will depend on the type of device used.
In the following paragraphs we will explain a procedure we have devices for the use of this application in future lines.

During the process of communication of the VARMAPP server with the smartphone, the server asks the mobile device the model of smartphone and access to a data base with the most recent models in the market to obtain the resolution of the device.

After that process, it is possible to realize the interpolation of the images to the screen size.

We have only implemented the module to change the images to a 800x480 pixels resolution as the smartphone use for the trials has this resolution.

In the resizing of the images, it has been necessary to use processing images libraries compatible with C#. One of the problems with working with this type of language is that Visual Studio 2010 doesn’t have libraries for image processing.

**Emgu CV [12]** is a cross platform .Net wrapper to the **OpenCV** [13] image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc. The wrapper can be compiled in Mono and run on Windows, Linux, Mac OS X, iPhone, iPad and Android devices.

In the lines below, we can see a section of the code that we have programmed in order to do the resizing of the images that we will receive form the Panoramio server.

```
Image<Bgr, Byte> img = new Image<Bgr, byte>(carpeta_fotos + photo_resize + ".jpg").Resize(800, 475, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC, false);
```

### 3.2.2 Effects

The first thing we need to create effects is to build a template pattern. We have created eight different effects for animating the transitions between the photograph. The main idea is to make final video more attractive to the end user.

The program access to the different effects through a random system that generates random images effects into the photographs.

The following chart shows the template pattern used to create the different effects.
The design of the template size has been calculated for a smartphone with a resolution of 800 x 480 pixels that is the resolution that has the device have use in the trials.

The number of pixels per horizontal division is 32. Summing them up we obtain the 800 pixels that correspond to the number of pixels in the screen of the smartphone.

The number of pixels per vertical division is 19. Summing them up we obtain 475 pixels. The screen of the smartphone has 480 pixels instead of 475 but we have used the 480 pixels value because it is better to work with whole steps and also it would be easier to program the effects in C#.

**Effect I**

The image begins to appear at the top left corner of the smartphone screen.

Fifty frames are used in this execution of this effect. As we are working with 25 frames/seg this effect will last for 2 seconds. At the end, the image freezes so the user can see the images still for a second. After that, a new image with a new effect will appear, this new effect will depend on the random program.
This and all the rest of effects have been created from scratch and programmed in C#.

Fig. 3.2 Effect I

Effect II
The image begins to appear at the left side of the smartphone screen.

As in the previous case, fifty frames are used in this effect. As we are working with 25 frames/seg this effect will last for 2 seconds. At the end, the image freezes so the user can see the images still for a second. After that, a new image with a new effect will appear, this new effect will depend on the random program.

Effect III
The image begins to appear at the top of the smartphone screen.

As in the case of effects I and II, fifty frames are used in this effect. As we are working with 25 frames/seg this effect will last for 2 seconds. At the end, the image freezes so the user can see the images still for a second. After that, a new image with a new effect will appear, this new effect will depend on the random program.

Effect IV
The image begins to appear at the top left corner of the smartphone screen.

Fifty frames are used in this effect. As we are working with 25 frames/seg this effect will last for 2 seconds. At the end, the image freezes so the user can see the images still for a second. After that, a new image with a new effect will appear, this new effect will depend on the random program.

After several trials, we have decided that fifty frames is an adequate value to produce a sense of movement in the images. Before continuing explaining the different effects, we are going to introduce a definition that will help us to a better understanding of the effects.
The Ken Burns effect is a popular name for a type of panning and zooming effect used in video production from still imagery. The zooming and panning across photographs gives the feeling of motion, and keeps the viewer visually entertained.

**Ken Burns effect I**

The image moves from the bottom right corner to the top left corner. This effect gives a sense of motion to the image.

Sixty frames are used in this effect. As we are working with 25 frames/sec, the effect will last for a bit more than 2 seconds. The reason why this effect needs more frames is because with a higher number of frames it is possible to obtain a sense of movement. Also, this effect needs a bigger resizing of the image to obtain the desired results. At the end, a new image with a new effect will appear, this new effect will depend on the random program.

**Ken Burns effect II**

The image becomes smaller and smaller. This effect gives the sense that the image is moving away from the viewer.

As in the previous case, sixty frames are used in this effect. As we are working with 25 frames/sec, the effect will last for a bit more than 2 seconds. The reason why this effect needs more frames is because with a higher number of frames it is possible to obtain a sense of movement. Also, this effect needs a bigger resizing of the image to obtain the desired results. At the end, a new image with a new effect will appear, this new effect will depend on the random program.

**Ken Burns effect III**

The image becomes bigger and bigger. That gives a zooming effect to the image.

As in Ken Burns I and II, sixty frames are used in this effect. As we are working with 25 frames/sec, the effect will last for a bit more than 2 seconds. The reason why this effect needs more frames is because with a higher number of frames it is possible to obtain a sense of movement. Also, this effect needs a bigger resizing of the image to obtain the desired results. At the end, a new image with a new effect will appear, this new effect will depend on the random program.

The first seven effects have been already defined. The eight and last effect is *special* as it depends on the user’s preferences. The viewer has to wear anaglyphic glasses to see this effect that will create a sense of 3D. Using this type of glasses, the user will be able to see anaglyphic images. In the next section we will explain what are anaglyphic images.
Anaglyphs

Anaglyph 3D [14] images contain two differently filtered colored images, one for each eye. When viewed through the "color coded" "anaglyph glasses", each of the two images reaches one eye, revealing an integrated stereoscopic image. The visual cortex of the brain fuses this into perception of a three dimensional scene or composition.

Plastic anaglyph filters produce an acceptable image at low cost.

Figure 3.3 illustrates the Anaglyph 3D process.

![Anaglyph Diagram](image)

**Fig. 3.3** Anaglyph

A pair of glasses with filters of opposing colors is worn to view an anaglyphic photo image. A red filter lens over the left eye allows graduations of red to cyan from within the anaglyph to be perceived as graduations of bright to dark. The cyan (blue/green) filter over the right eye conversely allows graduations of cyan to red from within the anaglyph to be perceived as graduations of bright to dark. Red and cyan color fringes in the anaglyph display represent the red and cyan color channels of the parallax displaced left and right images. The viewing filters
each cancel out opposing colored areas, including graduations of less pure opposing colored areas, to each reveal an image from within its color channel. Thus the filters enable each eye to see only its intended view from color channels within the single anaglyphic image.

**False 3D effect**

Before explaining how the 3D effect is generated, we will have a look at the 3D images that can be found in Panoramio and we will explain why we are going to use the false 3D in the application.

In Panoramio we can find some images that have been taken with two stereo cameras. That is why we consider interesting to incorporate the option of seeing 3D with conventional mobile devices thanks to anaglyph technology.

When having 3D images taken with two cameras, the disparity between two pixels depends on how far away the camera is from the subject.

It is impossible to generate a 3D effect accurately with one image without having information about the distance between objects.

False 3D is a technique which causes a series of images or scenes to fake or appear to be three-dimensional (3D) when in fact they are not. In order to achieve that, all pixels from the image are separated equally from each other. The result is a flat 3D effect, this means that we will see a flat frame in a determined position.

In order to generate a 3D effect using 2D images it has been necessary to program the server.

Before explaining the process in detail we will define the RGB concept that will help us to a better understanding of the matter.

**RGB**

The RGB [15] color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.
The first thing is that VARMAPP server does is to select the first image received from Panoramio.

![Fig. 3.4 Standard 2D image](image1)

Then, in the standard photograph (RGB color mode) the Green and Blue components are removed. RGB → Red=1, Green=0, Blue=0.

Red = 1, color red is maintained in the standard image.  
Green = 0, color green is removed from the standard image.  
Blue = 0, color blue is removed from the standard image.

The resulting image has only red color information, green and blue color information has been removed.

After applying the filter, the following image is obtained.

![Fig. 3.5 Filtered 2D image](image2)

Later, in the standard RGB photograph, the red component is removed.  
RGB → Red=0, Green=1, Blue=1.

Red = 0, red color is removed from the image.  
Green = 1, green color is maintained.  
Blue = 1, blue color is maintained.
The resulting images have green and blue information, red information has been removed.

After applying the filter, the following image is obtained.

![Filtered 2D Image](image)

**Fig. 3.6** Filtered 2D image

Finally, the two filtered images are combined.

Figures 3.7, 3.8 and 3.9 show a combination of the previous steps.

![Red Photograph](image) ![Cyan Photograph](image)

**Fig. 3.7** Red photograph  
**Fig. 3.8** Cyan photograph

![Resulting Photograph in False 3D](image)

**Fig. 3.9** Resulting photograph in False 3D

*All the photos used in this example have been taken from different tests realized during the execution of this project.*
In order to achieve the desired effect, it is important not to superpose the filtered images. They must be slightly displaced from each other. After several trials, we have decided that 60 pixels is the more adequate displacement number for watching 3D effect.

It is important to have in mind that sometimes the number of pixels will differ from 60. The number of pixels depends on the focal length used in the taking of the images. Different focal lengths will need different number of pixels, there are specific formules to calculate the number of pixels.

Different options in the number of pixels and displacement have been used in the trials but, as the 3D effects are not real, the fixed distance of 60 pixels has been enough in all cases.

It is important to clarify that the effect obtained with False 3D is not exactly the same as the effect we would obtain with stereoscopic images. False 3D is a technique which causes a series of images or scenes to fake or appear to be three-dimensional (3D) when in fact they are not.

### 3.3 Video codec

The program that fits our video editing requirements better is Avidemux.

Avidemux [16] is a free open-source program designed for multi-purpose video editing and processing, which can be used on almost all known operating systems and computer platforms. The figure 3.10 shows the program graphic interface.

![Avidemux graphic interface](image)

**Fig. 3.10** Avidemux graphic interface
Several video codecs can be found in Avidemux. The code that we have selected for the project is the video codec MPEG-4 ASP (Advanced Simple Profile)(Xvid) as it is one of the best compressors systems. Thanks to this codec, the program can be reproduced in almost all devices without an adapting process. The video exit selected is also in MPEG-4 format.

In order to achieve a sense of motion and good quality in the video we have used a 25 frames/second rate.

In the following lines we will analyze some ideas designed for generating the videos.

We have created a video header that will appear when running the application.

The following figure shows the video header in the smartphone.

![Fig. 3.11 Video header](image)

We have also created an ending message that will indicate that the video has finished.

The following figure shows the ending message in the smartphone.

![Fig. 3.12 Ending message](image)

We have also added music to all the videos for the viewer to have a more enjoyable experience.

In this application all videos will have the same music because as it is a prototype we have not gone in this topic in any depth.
One of the ideas for the future of this application is to give the user the option to select the kind of music that he wants to hear in the video or even insert the music that he prefers.

In the following section, we will explain how to develop the script so that VARMAPP server and Avidemux can generate a video.

3.3.1 Script

This section will explain briefly how the different parts that constitute the script are built. For a better understanding, we will use a simple example of a trial from our project.

Script route Castelldefels – Gavá

Parameters of the script

Header

Identifies the files as an Avidemux script.

```
//AD  < Needed to identify
//-- automatically built --
//-- Project: C:\Users\Faus\Desktop\tester4
```

`var app = new Avidemux()` Initiates a new request to Avidemux.

Video

All the images used by the script are loaded in this section. The images have been listed previously and stored in a file called *Photos* that is inside the VARMAPP server.

`app.load("C:/Users/Faus/Desktop/fotos/0.jpg");`

Segments

`app.addSegment(0,0,3932)` indicates the total number of images that constitute the video of the route Castelldefels-Gavá.

`app.markerA=0` indicates the number of the first image.

`app.markerB=3931` indicates the number of the last image.
Postproc

Post-processing parameters are established in this section.

\[ app.video.fps1000 = 25000 \] establish the speed of the frames. Resulting in a video rate of 25 frames per second.

**Video Codec conf**

\[ app.video.codecPlugin(“0E7C20E3-FF92-4bb2-A9A9-55B7F713C45A”, “mpeg4”, “CQ=4”, “(null)”) \]

Indicates that the video is coded in MPEG-4 and Constant Quantiser equals 4 (by default).

**Audio**

\[ app.audio.load(“MP3”, “C:/Users/Faus/Desktop/nombre_canción.mp3”); \]
\[ app.audio.codec(“Lame”, 128, 20, “80 00 00 00 00 00 00 00 00 01 00 00 00 02 00 00 00 00 00 00 00 00”); \]

Indica que el audio is in MP3 format with a bitrate of 128 kb/s.

\[ nombre_canción.mp3 \] the song that will sound when the video is displayed.

\[ app.save(“video.mp4”) \] that is the name of the video.

\[ app.Exit() \] indicates that the script finish at that point.

The VARMAPP server runs the script. When this process starts, Avidemux generates a video that is sent to a folder named videos that can be found in the SD Card of the smartphone. Once the video is saved in the folder, the smartphone will reproduce it automatically.

As it has been said in chapter 2, the VARMAPP uses a maximum of 5 images for POI. That fact will have an effect in the duration of the video.

\[ /mnt/sdcard/videos → video.mp4 \] route in which received videos are saved.

It is important to notice that a writing permission is necessary to let the application write to an external memory card on the device. In order to achieve that, the following data will be added in the AndroidManifest.xml.

<uses-permission> “android.permission.WRITE_EXTERNAL_STORAGE“
3.3.2 Testings

Several trials have been realized during the development of the project. In this section we will analyze the result of the most relevant trials. The number of images contained in each video depends on the route selected by the user.

This number is determined by the number of POI’s of each route and how many images are there. As it has been seen in chapter 2, the VARMAPP server selects a maximum of five images for POI. That fact will have an effect on the duration of the video.

The objective for the future is that this process could be automatically administered so the videos have a pre-establish duration.

The most relevant trial in the optimization of the application is explained in the following section.

**Optimized Codec**

The aim of this trial is to determine the difference of size between a video created with codec by default and a video created with optimized codec for obtains a bigger compression.

Those are the results after applying the changes:

Data for the creation of the video

Frames number: 2712 images.

Results

In both cases, the video duration is 1 minute and 48 seconds. That was the expected result as the duration depends on the number of images.

The results for the duration of the video are as followed:

<table>
<thead>
<tr>
<th>Non optimized</th>
<th>Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video size: 17.25 Mbytes</td>
<td>Video size: 6 Mbytes</td>
</tr>
</tbody>
</table>

The size of the optimized video is approximately 1/3 less than the size of the non optimized. That is a very interesting fact as small size videos occupy less space and can be transmitted at ever greater speeds.
**Trial route Castelldefels – Gavá**

Here we can see an example of route with the optimized codec.

After entering the route Castelldefels-Gavá in the VARMAPP application, 13 POI are obtained.

As the application uses 5 images for POI, the result is:

13 POI’s $\times$ 5 images = 65 images

The VARMAPP server will resize and list the images. Then, effects that create a sense of movement are applied to the images. Between 50 and 60 image repetitions are needed depending on the desired effect. In this case, the average value of repetitions will be 55.

65 images $\times$ 55 repetitions = 3575 images

It is necessary to add to this result the header and final video images. These images are common in all the videos and constitute a total of 352 images, 292 for the header and 60 for the final. Therefore, the final number of images used in the video of this route is 3575+352=3927. The exact number of images in the trial is of 3932, this slight variation is due to the use of the random system.

Once the process if finished and the video has been created, the following results are obtained:

**Results**

Video duration: 2 minutes and 37 seconds.

Video size: 7,795,313 bytes

Therefore, a 500 MB flat rate user could download and watch 62 videos using the maximum speed. This data is approximate as it depends on the route selected by the user and the number of POI’s that constitute the route.
The following graphic shows the different blocks that have been programmed in the VARMAPP server in order to administrate the process, from the sending of POI’s of the route to the VARMAPP server to the creation of the final video.

In this chapter, we have study the processing of images. In the next chapter we will analyze the environmental impact of the project and the future lines for the improvement of the application.
3.4 Summary

Summarizing this chapter, we can conclude that:

- We have learned how to use and code video program from the VARMAPP server though its script.

- We have realized that the duration time of the application (from the user request to the display of the video in the smartphone) is determined by Avidemux. Over all the processes in the making of the video, Avidemux is the slowest.

- We are satisfied with the results obtained in the different trials we have realized and also with all the things that we have learnt after finishing this chapter.

- For a future commercialization of the application it would be necessary an improvement of the software, substitution of the VARMAPP server for a real server. Also, we would have to do more trials.
CHAPTER 4. GENERAL CONCLUSIONS

This paper defines a new application for mobile devices that has been specifically designed for this project. An important fact to have into account is the innovation of this application as no other similar application can be found in the market at the moment.

We also think that this application is an attractive idea and could be successful from a commercial point of view.

A defined body of knowledge has been required in the realization of this project, it has been necessary to study how Android works, programming languages as C#, processing of images and video, coding and communication protocols between others.

The development of this project has been very extremely interesting and has allow us to study different programming aspects that haven’t been covered during the degree.

This is an application clearly defined from the very beginning, in which all functions and modules that compose the application has been deeply analyzed.

This final project is a demo in which the essential aspects are solved for a correct functioning of the application. Nevertheless, we are aware that some things could be improved for a better functioning of the application for its commercialization.

Despite the troubled and challenging times and the heavy workload, we are satisfied with the result obtained in this project. It has been a hard job to summarize the work of seven months in this report and write the project in English.

From the very beginning, it was clear that we wanted to build our application in Android. Therefore, we had to study how the Android system works and the necessary tools for a correct functioning of program.

As Android applications are quite a new field of research, several books and internet sources were consulted in the development of our project.

After finishing the Android section, it was necessary to design a communication architecture and protocol definition between the different devices used in the project.

Then, we realized that a server should be created. The programming of the server from the scratch has taken several hours of the project because we did not have experience in this field. The programming has been coded in C#.
We have learned about the functioning of the Google Maps and Panoramio servers and how they do requests. We have also learned how to read JSON files and how to extract the significant data for our application.

At that point, it was necessary to deal with the images obtained from Panoramio. In order to do so, we spent several hours learning how the image processing works.

The result of all this work is the creation of a new application called VARMAPP.

### 4.1 Environmental impact

The environmental impact of this project is minimal as it is an application for smartphones. The section that deals with the Android development system has been realized in our own laptop; therefore the only expense to bear in mind is the electric consumption.

One direct impact of this application is that it is executed in smartphones, therefore there is electromagnetic radiation, use of batteries, etc.

Our laptop has also been used in the design and programming of the VARMAPP server and in the editing video section. Once again, the environmental impact has been minimal.

In the case of bringing the application to the market, it would be necessary to use a “real” server or a set of rack’s if the commercial success is as expected. This server should be always working, so a study of electric consumption would be required.

The paper consumption is minimal as everything has been digitally implemented.
4.2 Future lines

Although the goals of this project have been achieved, new ideas for improvement arise.

Some of these ideas are listed below

- **VARMAPP**: design of a multi-platform application compatible con tablets.

- **Routes**: restriction of the distance in the user selected route. It would be nice for the user to have the option of selecting an area of interest in the route and create a video of that specific part. Also, it would be interesting for the user to see in the display or the route restaurants, shopping centers, places of interest, etc.

- **POI’s**: Notify of the presence of points of interest when detecting a large number of images in certain coordinates.

- **VARMAPP server**: use a real server for developing the application for a commercial use.

- **VARMAPP server optimization**: use of libraries specifically designed for image processing.

- **3D stereoscopic images**: updating VARMAPP so that the user can watch 3D videos.

- **Video**: An interesting thing would be the creation of a list at the end of the video. The list would contain information about the exact location of the images, the selected route and so on. Another idea would be to fix the duration of the videos.

- **Streaming**: this feature makes possible for the user to watch the video before it is completed.

- **Users**: it would be good to identify the users that are using the application so the server can send videos correctly.

- **IPv6**: adapt the application to the IPv6 protocol.

- **Version**: create a payment version with more functions.
BIBLIOGRAPHY

- Books:


- References

[1] Official Website Android
URL: http://www.android.com

[2] Official Website IDC
URL: http://www.idc.com

[3] Official Website Wikimedia Commons

URL: http://developer.android.com

[5] Official Website Oracle
URL: http://www.oracle.com/technetwork/java/javase/downloads/index.html

URL: http://developer.android.com/sdk/index.html

[7] Official Website Eclipse
URL: http://www.eclipse.org

[8] Official Website JSON
URL: http://www.json.org

[9] Official Website Google Developers
URL: https://developers.google.com/maps/documentation/directions/?hl=es

[10] Website ABC
URL: http://www.abc.es/20120125/medios-redes/abc-fotos-lugares-mundo-201201250955.html
URL: http://www.panoramio.com/api/widget/api.html

[12] Official Website Emgu CV

[13] Official Website Open CV
URL: http://opencv.willowgarage.com/wiki

URL: http://en.wikipedia.org/wiki/Anaglyph_3D

URL: http://es.wikipedia.org/wiki/Modelo_de_color_RGB

[16] Official Website Avidemux
URL: http://avidemux.org