



INTERNSHIP REPORT PATTERN RECOGNITION UNISA ITALY

BRETON-BELZ Emmanuel - 2nd Year Internship

Une grande école pour réussir

ENSICAEN
6, boulevard Maréchal Juin
CS 45 053 – F- 14050 Caen Cedex 4
Tél. +33 (0)2 31 45 27 50
Fax +33 (0)2 31 45 27 60

Thanks

I had like to thank Mario Vento for hosting me in the labs, give me the oportunity to see conferences of foreign coomputer science doctors and following me during the intership.

I thank Pierluigi Ritrovato who gave me my subjects and guided my to my targets and answered to my questions.

Thanks to Hugo Descombes too, who helped me during the first part of my internship.

Table of contents

1	Introduction	1
2	Lab and place	2
2.1	The MIVIA Laboratory	2
2.2	Place and life in Italy	3
3	Evaluation of the need	4
3.1	Helmet HC1 needs	4
3.2	The software needs	4
4	Studies	5
4.1	Helmet study	5
4.1.1	Helmet itself	5
4.1.2	Embedded systems	5
4.1.3	Cross-compilation	6
	Description	6
	Process	6
	Toolchain	6
4.2	Pattern recognition study	7
4.2.1	OpenCV	7
4.2.2	Existing code	7
	Key points	7
	Descriptor	8
	Homography	8
5	Achievements	9
6	conclusion	10
7	Annexes	12

Chapter 1

Introduction

During the second year of engineering school at ENSICAEN we have 3 month internship that I decided to do in the MIVIA Lab of Salerno's university in Italy. They are specialised in image synthesis which is my major course, moreover I am interested in a double diploma in this university which gives me the opportunity to see the place and teachers, learn the Italian and make some contacts.

My internship subject has been split in two. First a study on the Motorola HC1 helmet used for example in military domain. The aim was to see if we could make the helmet compile Linux for helmet. Then develop an application capable of recognise patterns in an image and store the position, the number of patterns found and the type of each pattern. So as to determine the name of a global structure which the patterns are components.

In this report I will present the laboratory and the place of the internship, the study of the hardware for the helmet and the software (OpenCV mainly) for the application. What exists and where my projects are situated in their environments. After that I explain what I tried to solve the problems and what are the results.

Chapter 2

Lab and place

2.1 The MIVIA Laboratory



for Macchine Intelligenti per il riconoscimento di Video, Immagini e Media which means intelligent machines for video, images and media recognition. The laboratory is located in Fisciano, Campania, Italy, near Salerno and Napoli as we can see below.



Figure 2.1: Location of the University

As its name suggests, except teaching computer science, doctors of the lab work on pattern recognition, classification, media analysis and many parallel subjects like autonomous drones and robot vision.

Because of the opportunity of double diploma, the proximity with France, the possibility to learn a new language even if I know that everybody speaks English in lab. I was really optimistic about the place and the laboratory.

2.2 Place and life in Italy

The first month of internship, I lived in Carpineto, 25 minutes to the university by foot. It was envying the view on the mountains in the morning and the evening but when I wanted to move the week-end, buses didn't pass near my flat so that I had easily hours of walk to get back home. I moved on to Salerno, in the centro storico. Then it was easier to move the week-ends.

The main tourstic place around is the Amalfitan coast which is really beautiful. There is also Napoli and Pompei which are really near. There is also a lot of beaches and montains. A really nice place where you can have snow in the winter and 33 degrees during weeks in the summer.

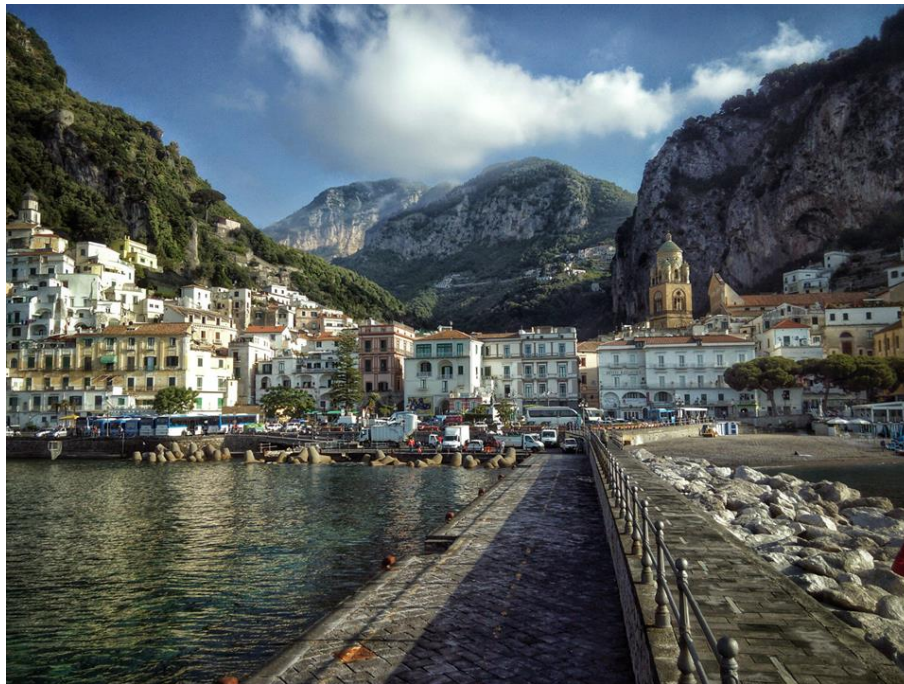


Figure 2.2: Picture of Amalfi early in the morning from the harbour

In the picture 2.2 the beach is not set for summer so there is nowone, but in july, it becomes overcrowded because of tourism. Also, I found that this part of Italy is really human. It feels like you can trust people really quickly because they are really sympathics. In that way, there is few offical contract. That is suprising when you come from France and it requires a thousand of documents to get a flat. Here, the ID card is enough. People are always happy but a bit slow when the walk that is getting on my nerves when I am in a hurry. I took advantage of the intership to make a lot of trekking in the amalfitan coast and around Fisciano. Visite ruins, Napoli and see a lot of churches because christianism is really present in Italy.

Chapter 3

Evaluation of the need

3.1 Helmet HC1 needs

The HC1 Helmet from Motorola is a professional uses helmet dedicated to wide or site conditions, that is why it costs around 3000\$ with the camera. It is used to show images in augmented reality in the little screen in front of the left eye. Full voice commanded, it uses a embedded version of windows which was the real problem. You can look at the figure 3.1 to have an idea of what it looks like.

Despite the fact that the Helmet is thought to add fonctionnalities, a client of the MIVIA Lab asked them if it was possible to install Linux or Android on the helmet. Even if they loose the voice command system and the drivers to run the camera etc. They asked me to try, at least, to do it.



Figure 3.1: Picture of the HC1 helmet with the camera plugged

3.2 The software needs

At work, when people have to make maintenance of the material, they encounter a problem with the density of the maintenance manuals which can make around 700 pages. We try to ease the maintenance by recreating manuals that focuses on the material that the technician is looking at. To obtain this result, we have to analyze the images of the camera and extract the type of material. That is what the laboratory asked me to do.

This solution can apply to a lot of other objects to find monuments and extract their description for exemple.

Chapter 4

Study of the existing

Once again this part will be splitted in 2 because the subjects are totally different. I am going to talk first about the helmet, that required 1 month of test and studies. I will explain the procedures in the next chapter.

4.1 Helmet study

4.1.1 Helmet itself

I began my study by acquire some knowledges about the helmet itself. It has been release in the end of 2013, build for harsh condition, its price makes it unaccessible for the public. The army and building companies sow a good opportunity in this technologie to ease the work by bringing communication into the field.

The helmet is equiped with a batterie, Wi-Fi and bluetooth connections, a camera and it can be wear under the work helmet. Everything is voice commanded and very responsive thanks to Motorola's work. Windows CE 6.0 is used on the last release of the helmet image. It is good for the next section to understand that the booting system and the update system of windows CE are related. That means that you can change the file system for an update but it is windows itself that validate and copy the files on the intern memory from the SD card.

4.1.2 Embedded systems

I learned a lot about embbeded system, mainly on the boards and all the materials related to it. In our case, the materials inside the helmet are not know and not published on the Internet. Pierluigi contacted the company that brought us the helmet but they couldn't tell us which board was used in the helmet. I must have guessed which TI technology it was because the datasheet reference a TI OMAP 3 microprocessor.

That is mainly why I put my effort on the "ISEE – IGEP COM MODULE" built in with a TI OMAP 3 processor. It was at the top of the art when the helmet released and the smallest board with this processor. It corresponds well with the size of the hardware

slot. In any case, if the linux kernel is compatible with the processor a cross compiled file system should boot and at least it should show an image on the screen.

4.1.3 Cross-compilation

Description

The cross-compilation is compilation for a different achitecture than the architecture that makes the computations. In the figure 4.1.3 you can see that the source code can be compiled for different architectures. The aim here was to compile a Linux kernel compatible with TI OMAP 3 processors from an Intel x86 machine.

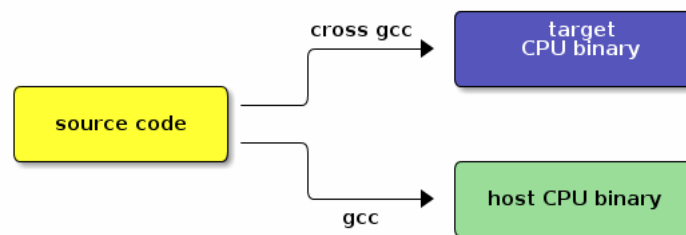


Figure 4.1: Cross compilation and compilation difference

Process

The cross tool chain is capable of compile a Linux kernel for a lot of achitectures. Of course it requires a long time to compile the toolchain and then cross compile the kernel to get the binaries but it is cost effective. The time nessessere to compile the toolchain and the kernel on the TI OMAP 3 would be longer.

Toolchain

A toolchain is a set of distinct software development tools that are linked (or chained) together by specific stages such as GCC, binutils and glibc (a portion of the GNU Toolchain) [2].

A toolchain requires binutils such as assembler and linker, that produces the binaries. Also compilers for deferent languages like C, C++, Java etc, that transforms any language into another. A C library to gain access to kernel calls and a debugger that can be used or not during the compilation. We can see well on the diagram 4.1.3 from (avrfreaks.net's forum) where each composant is located :

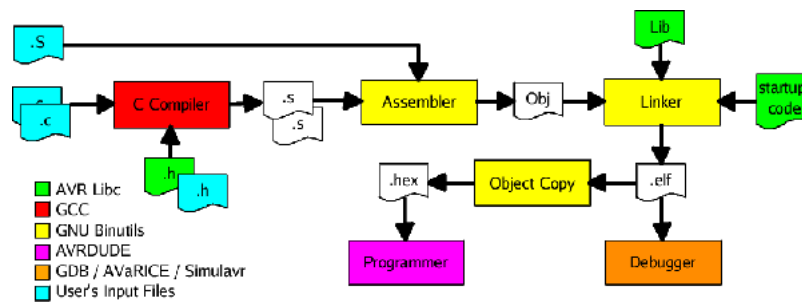


Figure 4.2: Cross compilation and compilation difference

4.2 Pattern recognition study

4.2.1 OpenCV



OpenCV is an image analysis and synthesis library that brings all the necessary for video and photo analysis. Introduced in 2000, it is now used a lot in the fields that require image analysis. It is built around a modular system that allows users to install the library depending on their needs. Except the non free module, the library is licensed BSD which means that we can reuse and modify all the components freely. In this document I will talk more about precise modules of OpenCV as a base and features2d modules.

4.2.2 Existing code

To get some examples of code and see how it works I simply looked at the OpenCV tutorials that are available on the internet. For example check [here](#). It brings a simple way to extract a pattern and make the homography to draw the corners with `lign` OpenCV function.

Key points

SIFT points are points of interest in the image. They give the position of an area where all the pixels have approximately the same properties. Using the Laplacian of Gaussian of the image, maxima and minima which correspond to these areas can be extracted.

The biggest advantage of these points is the invariance in transformation like rotation, scaling and translation. Pattern detection algorithm based on this point is insensitive in term of scale and rotation. That is mainly why it has been chosen for this application.

As a result, by the time that the resolution of the pattern is over an acceptable threshold. We can move and rotate the object without altering the detection.

SIFT Descriptor

The descriptors are computed from the key points. The descriptors associated with the SIFT key points are locally oriented histograms around a SIFT key point [4]:

- We divide space around each key point (x, y) in N^2 squares of 4 by 4
- We compute the gradient $G_x(a, b, \sigma)$, $G_y(a, b, \sigma)$ for the 4 by 4 by N^2 points (a, b)
- For each 4 by 4 square, we compute a histogram of the orientations in 8 directions, multiplying by: (1) the module of the gradient (2) the inverse of the distance to the key point (x, y) .
- To be invariant in rotation: the local orientation of the key point $\theta(x, y)$ is used as the origin (null orientation) of the histograms.

Homography

Just to give a simplified idea, the familiar Cartesian plane is composed by a set of points which have a one-to-one correlation to pairs of real numbers, i.e. X-Y on the two axis [1].

In our case, the points are the key points extracted. By extracting the homography matrix, we can transform the corners of our image to 4 other points representing the position of the object on the scene [3].

On the image behind we can see that the homography doesn't depend on the corners. All the key points can be a base to analyze the transformation. And using the perspective transform of OpenCV that multiplies the corners by the homography matrix. We obtain the position of the corners on the captured image.

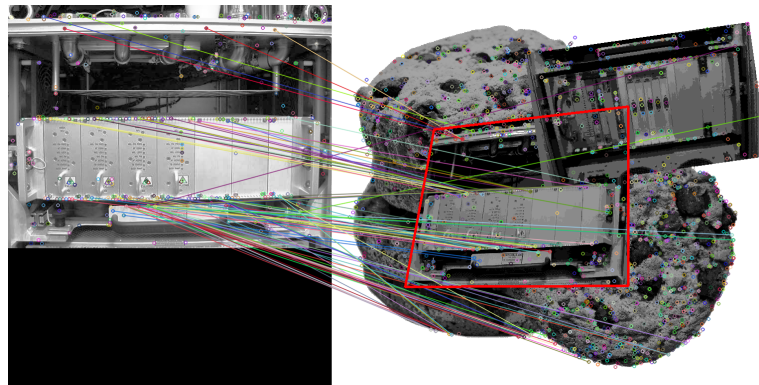


Figure 4.3: Homography with a corner hidden with openCV

Chapter 5

Achievements

This part will a melting pot of issues and success and how I found out how to correct them if I could. I will gather the installations for OpenCV and the cross toolchain because the main idea is the same. Then I will spit the technical part in two as usual otherwise it will not be clear enough.

5.1 Installation and tests

Chapter 6

conclusion

Bibliography

- [1] CorrMap. The homography transformation. http://www.corrmap.com/features/homography_transformation.php, 2013.
- [2] elinux.org. Toolchains. <http://elinux.org/Toolchains>, 2015.
- [3] D.J. Fleet & A.D. Jepson M. Brubaker, S. Mathe. Tutorial: 2d homographies. <http://www.cs.toronto.edu/~jepson/csc2503/tutorials/homography.pdf>, 2004-2007.
- [4] Antoine Manzarena. Cours etasm. http://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCIQFjAA&url=http%3A%2F%2Fwww.ensta-paristech.fr%2F~manzaner%2FDownload%2FEPPF_Bucarest2006%2FCours_EPF%2FChapitre3.ppt&ei=K4ujVdrjMcOssgHHg4bYDw&usg=AFQjCNEgM2Usx3DKDfBQbo3dXfu6dQ2nzw&bvm=bv.97653015,d.bGg, 2006.

Chapter 7

Annexes

OpenCV tutorial : http://docs.opencv.org/doc/tutorials/features2d/feature_homography/feature_homography.html#feature-homography

List of Figures

2.1	Location of the University	2
2.2	Picture of Amalfi early in the morning from the harbour	3
3.1	Picture of the HC1 helmet with the camera plugged	4
4.1	Cross compilation and compilation difference	6
4.2	Cross compilation and compilation difference	7
4.3	Homography with a corner hidden with openCV	8

Summary