ENSICAEN
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE

Internship report pattern recognition UNISA Italy

2nd Year Internship

- Emmanuel Breton-Belz
- 2nd year Computer science
- University year 2014-2015
- UNISA supervisor : Mario Vento
- ENSICAEN superviser : Luc Brun

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, giving me the opportunity to see conferences of foreign computer scientists and following me during the internship.

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

I Thank Hugo Descombes too, who helped me during the first part of my internship.

Finally, I want to thank Allessia who contacted me for the papers and gave me good advices on the matching system, the international relation departement of ENSICAEN which prepared all the documents for ERASMUS facilities.

Table of contents

Chapter 1

Introduction

During the second year of engineering school at ENSICAEN we have to realize a 3-month internship that I decided to do in the MIVIA Lab of Salerno's university in Italy. They are specialized 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 divided in two. First a study on the Motorola HC1 helmet, used, for exemple, in the military field. The aim was to see if we could compile Linux for the helmet and install it on. Then, develop an application capable of recognizing 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

Except teaching computer sciences, doctors of the lab work on pattern recognition, classification, media analysis and many parallel subjects like autonomous drones and robot vision.

The laboratory is directed by Mario Vento. He is assisted by Alessia Saggese. Pasquale Foggia, Gennaro Percannella and Pierluigi Ritrovato (who is also my tutor). They all are teachers and researchers in the lab. There are also several students and graduated that work on thesis in the laboratory. I was more in contact with Antonio Greco and Raffaele Iuliano. All students and graduated was working on different projects

and startups and the laboratory itself works for and with international companies.

You can find all of this information on their website at this address : `http://mivia.unisa.it/`

Because of the opportunity of double diploma, the proximity to France, the possibility to learn a new language even if we spoke English between us. 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. I was enjoying the view on the mountains in the morning and the evening, but when I wanted to move during the weekends, 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 old town. Then it was easier to move the weekends.

The main touristic place around is the Amalfitan coast, which is beautiful. There are also Napoli and Pompei, which are respectively a big harbor and internatialy knows a touristic place. Globally, it is 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 of Amlfi there is nobody, 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 sympathizing. That is surprising when you come from France and it requires a thousand of documents to get a flat. In Salerno, the ID card is enough. People are always happy, but a bit slow sometimes. I took advantage of the intership to make a lot of trekking in the amalfitan coast and around Fisciano. Visit 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 use helmet dedicated to wide or site conditions. 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 an embedded version of windows which was a 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 functionality, a client of the MIVIA Lab asked them if it was possible to install Linux or Android on the helmet. Even if they lose 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 example. To do that I had to use OpenCV pattern detection via SIFT points matching.

Chapter 4

Study of the existing

This part will be divided in 2 because the subjects are totally different. I am going to talk first about the helmet, that required a bit less than 1 month of testing and studying. I will explain the procedures in the next chapter.

4.1 Helmet study

4.1.1 Helmet itself

I began my study by acquiring some knowledges about the helmet itself. It has been released at the end of 2013, build for harsh condition, its price makes it inaccessible for the public. The army and building companies saw a good opportunity in this technology to ease the work by bringing communication in the field.

The helmet is equipped with a battery, Wi-Fi and Bluetooth connections, a camera and it can be worn under the work helmet. Everything is voice commanded, and very responsive thanks to Motorola's work. Windows CE 6.0 is used in the last release of the helmet image. It is good for the next section to understand that the starting 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 embedded system, mainly on the boards and all the peripherals. In our case, the components inside the helmet are not known 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 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 in the state 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[1] for a different architecture 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.

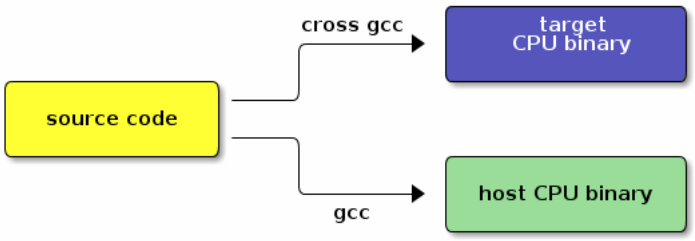

Figure 4.1: Cross compilation and compilation difference

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) [**?**].

A toolchain requires binutils such as assembler and linker, that produces the binaries. Also compilers for different 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 component is located :

Process

The cross tool chain is capable of compiling a Linux kernel for a lot of architectures. 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 necessary to compile the toolchain and the kernel on the TI OMAP 3 would be longer.

[1] Process that generates a binary file for a specific architecture.

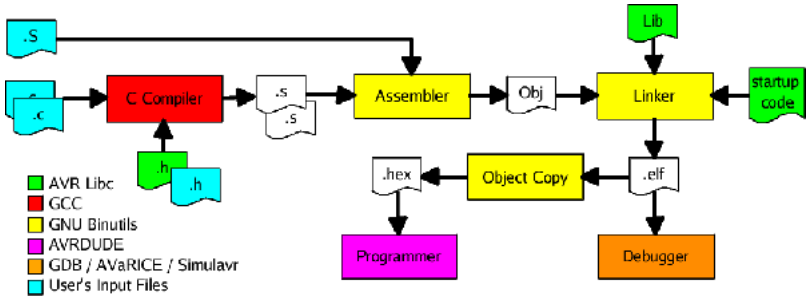

Figure 4.2: Compiler process and files

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 it in the annex section. It brings a simple way to extract a pattern and make the homography to draw the corners with the 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 corresponding to these areas can be extracted.

The biggest advantage of these areas is the invariance in transformation like rotation, scaling and translation. Pattern detection algorithms based on SIFT points are insensitive in term of scale and rotation. That is mainly why it has been chosen for this application. By the time that the resolution of the pattern must be 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 [?]:

- 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 squares, 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 [?].

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

On the image behind we can see that the homography doesn't depend on the corners. All the key points can be a base for analyze. 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.

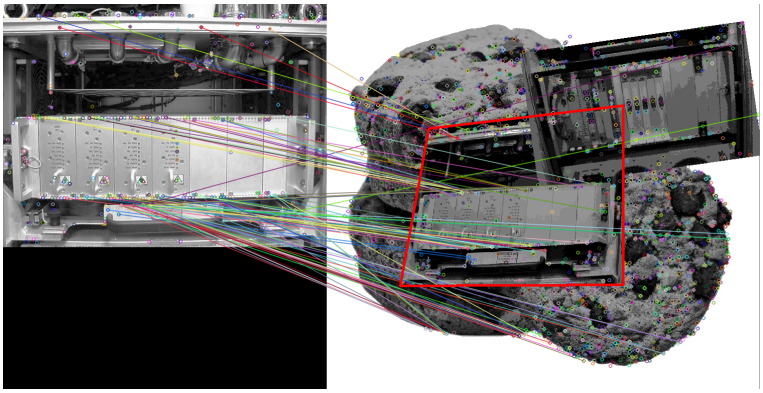

Figure 4.3: Homography with a corner hidden with openCV

Chapter 5

Achievements

I tried to apply agile methods during the development of the application. I split each part in daily tasks and I reported it the day after if they were not done. When I spent more than three days on a single task I informed Pierluigi and I looked for help in the lab. It worked really well and I can tell know that if works even on a project where I was single. This method matched with my rhythm and my progression.

In the figure 5.1 we can see all the progression of the internship (all the tasks are explained in the following pages). I set two important meetings for the project, but there were meetings with Mario Vento for global satisfaction and progression feedbacks that I did not mention. Globally, I think my time was well organized and I had a regular progression.

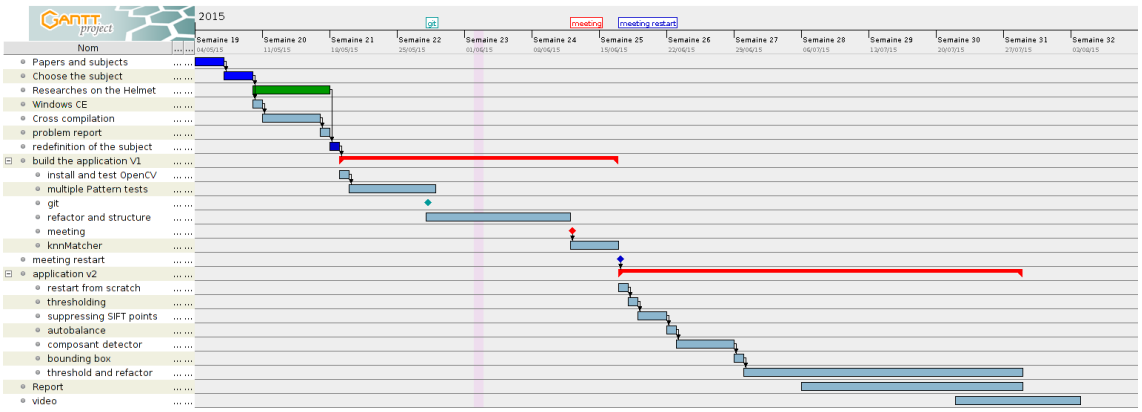

Figure 5.1: Gantt diagram

5.1 Installation and tests

For keeping the idea of two distinct projects, this part will turn around on issues and successes. I will explain how I 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 divide the technical part in two as usual, otherwise it will not be clear.

5.1.1 Helmet

Cross GCC

Cross GCC is pretty easy to install, you have to get the archeve and compile the cross compiler for your architecture. First of all I saw in that the tutorial required an Ubuntu 12.10LTS[1] at least. I used virtual box to get a virtual machin[2] of this distribution on my windows operating system. Next, I simply followed the tutorial that explains how to install the cross compiler. This part was really easy, after have downloaded GNU cross G++ I ran the commands and the compiler was installed.

Kernel compilation

The compilation of the kernel is almost the same as the compiler, except that you compile for another platform [3]. Of course the compiler handles ARM cortex A8 for TI OMAP3. I just had to check the option for this CPU[4] during the installation and then the kernel I compiled was compatible with this architecture. As a result, I got a file system ready to boot on an ARM device.

Tests

Once the file system is compiled, I had to put it inside the helmet memory. Two main problems remained, first the helmet memory was inside the helmet itself and not in the memory stick. And secondly, you can only copy files using Windows CE on the intern memory. This means that you cannot erase Windows CE file system and replace it by another.

That is where I found out that the project was compromised. I had few knowledges in Windows CE and the fact the helmet was not started in the first place meant that I had to repair this Windows CE first. Moreover the interface is voice commanded. I estimated the time to solve the problem longer than my internship.

Thus, I tryed to launch the file system on qEmu which failed. When I explained all of that to my superior they decided to change the project. Still, I had to send a little report to Mario Vento with the different problems that I encountered during the project.

[1] Long Lime Support of Linux based operating system.
[2] Software to emulate operating systems
[3] Here it is the OMAP3.
[4] Central processing unit.

5.1.2 Pattern scanner

OpenCV installation

I installed OpenCV 3.0 which is the last version and tried it. It has a base for basic image applications and modules that you can install by adding them to the module folder and compile them. Unfortunately, one module required to perform the test was not working with the 3.0 version. It is the non free module. I tried several times, but I finally decided to switch back to a previous version. After uninstall everything and install the 2.4.9 version. The module was functional. I tryed the code of the tutorial I quote earlier and I had the first pattern recognition result. That took me 4 days in total including the eclipse configuration that require adding all the libraries in the configuration of the compiler one by one.

First build on eclipse

As I just said, to make it work with eclipse I had to set the configuration for OpenCV project, otherwise It does not compile. Right click on the project, go to properties, than go to the settings of the compiler (GCC C++) and select includes. There add the path to the local install folder where the OpenCV's includes are located. After that you have to set the libraries to link one by one. In the same window goes in libraries section and add their names like opencv_core... Add the search path of the local lib folder. Then save the setting and compilation should work.

5.2 Main problems

5.2.1 Related to the helmet

I had several issues before I began working on the helmet. Some of them are related to the hardware and others to the operating system.

Hardware

The first problem we encountered was getting the component references of the helmet like processor, camera, mother board, etc. Pierluigi, my tutor sent mails to the company which proposed the project. Unfortunately, they were not able to gain access to this information. I looked for it on the manuals and documentation that Pierluigi gave to me and the Internet. Every time, I could not get the precise references, only the names of the product are available.

Besides, there was little issues with the helmet itself because the alimentation module was missing. That module allows us to plug the helmet directly to the outlet. To resolve the problem I had to load the batteries and plug it to the helmet to boot[5] it.

[5]Start the operating system.

Emulator

I download and easily install the emulator (qEmul) advised in the tutorial that I followed but I couldn't run the operating system on it. The operating system started with a black screen and then nothing happend. I did not find a solution to this problem because I found out that the project was already compromised by the fact that we did not have access to the references of the components. Still, I was a little bit surprised that the emulation was not working because all the steps of the tutorial was working except this one.

The operating system

Window CE 6.0 was made for embedded systems to concurrence Linux. The problem with the helmet was the fact that the system could only be updated and not erased. For example, if somebody wants to replace the file system, he has to put on an SD card, an update of windows that is compatible. This update would be written on the intern memory to replace the actual system. Unfortunately, except Microsoft I do not know if somebody can do this which means that Linux will not work. Perhaps there are other ways to do it, but I do not have that kind of skills. Thus I looked for help on the Internet, but Windows CE does not have a lot of dedicated forums.

5.2.2 Related to the software

The matcher

After 3 days of stagnation, making some tests and learn about the matching system. Alessia, the thesis companion of Mario Vento made me realize that I was on the wrong way using the match function of OpenCV. In fact, I could not find multiple patterns with that method because it selects the nearest descriptors[6]. The result was that I had to launch several times the algorithm to get new matches again and again. I found the knnmatcher that perform a full matching and keeps the N best matches. That was not enough so I switched to the radius matcher that need a threshold parameter. That is finally how I introduce the threshold in the program. This value depends on the object, the distance and the conditions of the scene. In another way, the matcher keeps the matches which differ less than the threshold.

The algorithm returns a vector[7] of vector of DMatches[8] so I put it in a simple vector of DMatches and it gave me very good results. The algorithm was able to find multiple times the same match on the scene. In the figure 5.2 we can see the fact that every time the pattern is represented it has matched it. Not just the best ones which here corresponds to the first one on the left.

[6] Those which have a minimum hamming distance likeness
[7] Collection of the standard library in C++.
[8] A structure representing each match.

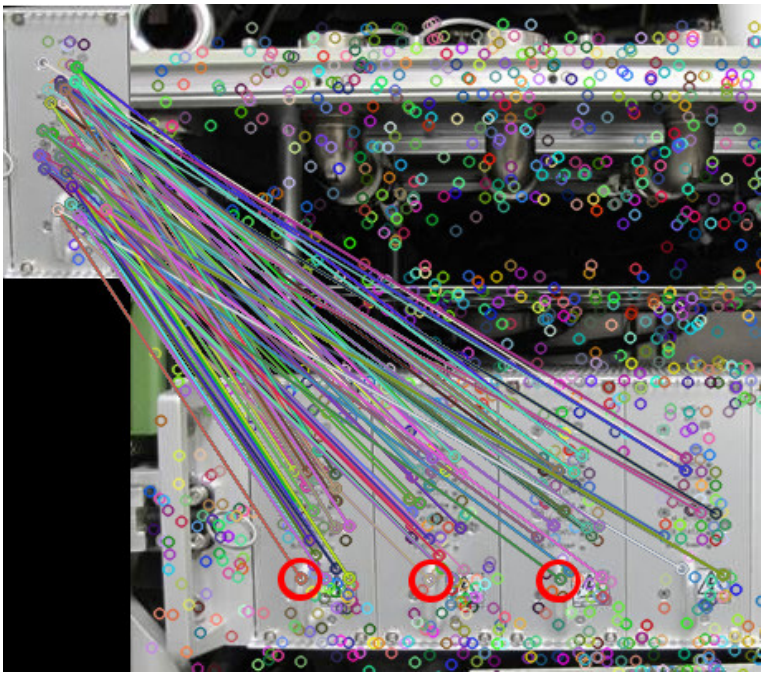

Figure 5.2: Simple draw of radius match with 3 times the same match surroundded in red

Get many time the same pattern

With Pierluigi, we agreed on two methods. The easy one was to draw in black the image and then make the homography another time. The hardest one was to look for the key points and descriptor inside the corners[9] and erase them of their containers. Then do the homography again to get another pattern. I began with the second one that sounded more relevant.

To be clear, I have to present the DMatch structure that is detailed in the annexes. The important attributes are the train index and the query index. They represent the fact that a descriptor has been found in the two images. But only the index of the descriptor is stored.

I first thought I could easily get the points that gave me the corners after the homography. In fact, I needed to remove the points that concern the pattern that has been recognized by the homography. There was no other way but to extract all the indexes of the points inside the corners I just found and remove them from the vectors. By chance, the homography takes in parameter vectors of points describing the positions of the descriptors (and the key points by the same way).

On the first version of this algorithm I was looking if the point was in the bounds represented by the 4 segments of the rectangle of the pattern found. I found out when

[9] Those we just found.

I was rotating the image that the points was removed the wrong way because the shape of the pattern was a lozenge. A lot of points were considered out of the bounds and the other that had nothing to do with the pattern was removed.

I found on the internet, a method [?] that provides the answer of my problem in O(n)[10]. This method finds the number of edges that a line from one of the points to the infinite crosses. If it is odd then the point is inside the polygon, otherwise it is outside. The picture 5.3 illustrate the idea.

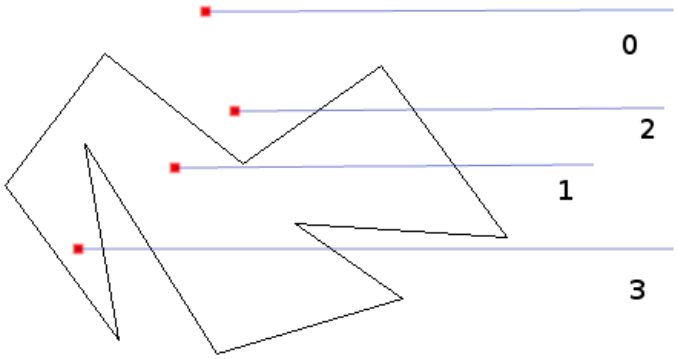

Figure 5.3: Illustration of the algorithm which tell if a point is inside a polygon

Corners

OpenCV does not implement a structure for the corners. The corners correspond to the vertices of the polygon that represents the pattern. I isolated them in a structure because that was a good proof of the fact that the algorithm found efficiently a pattern. It also helped me to find out that the first version of the algorithm which removes the points was malfunctioning.

I let the structure as it is because I told myself that, in the future, the application could need the corners of the pattern to compute move analysis for example.

Others

My biggest parallel achievement is an algorithm that helps the user of the application to find the best threshold for his patterns. It depends on the number of patterns you want to recognize because the threshold must be higher for certain conditions and, more there is a pattern in the image, the more there are different conditions.

The algorithm takes a scene and a pattern in input plus the number of times the pattern is represented in the image. The result of the algorithm is the first threshold

[10] Complexity of the algorithm here n represents the number of points in the container.

that got all the patterns in the image. It computes a range of 150 values, that is why I can take several seconds to compute.

```
manu@manu:~$ cd workspace/Scanner/Debug/
manu@manu:~/workspace/Scanner/Debug$ ./Scanner launcher_test MIU 4 test
base value threshold : 200
base value minHessian: 400
init done
opengl support available
component : MIU.jpg
Corners :
(1174.82,444.884) - (1161.82,540.462)
(995.836,418.483) - (980.346,513.714)
component : MIU.jpg
Corners :
(1189.17,345.908) - (1174.21,449.613)
(1009.87,322.949) - (989.461,427.661)
component : MIU.jpg
Corners :
(1216.4,166.815) - (1200.3,255.987)
(1033.75,147.362) - (1019.67,237.496)
component : MIU.jpg
Corners :
(1197.68,259.142) - (1189.82,354.518)
(1023.08,233.946) - (998.239,332.142)
```

Figure 5.4: Screenshot of the algorithm finding four times the MI unit pattern

We can see here that for units has been found with the positions of their corners.

I also had to create a video for Pierluigi to showcase the possibilities of my application. Therefore, I used VLC media player to stream my screen during a demonstration. I turned the stream into mp4 and used Kdenliv to make some modification and add titles. When the result was good enough, I used GNOME Subtitles to write a .str[11] file that corresponded to the video. Then I uploaded the video on Youtube to share it with Pierluigi. I made some changes in the beginning of the video several times. That took me a long time because I had to change manually all the subtitles that passes after the added or suppressed section.

The video is accessible here : `https://youtu.be/q23RjtwAqDU` you can see in the figure 5.5 the page where I edited the properties and added the subtitles to make it available on Youtube.

[11] .str is the standard format of subtitle files.

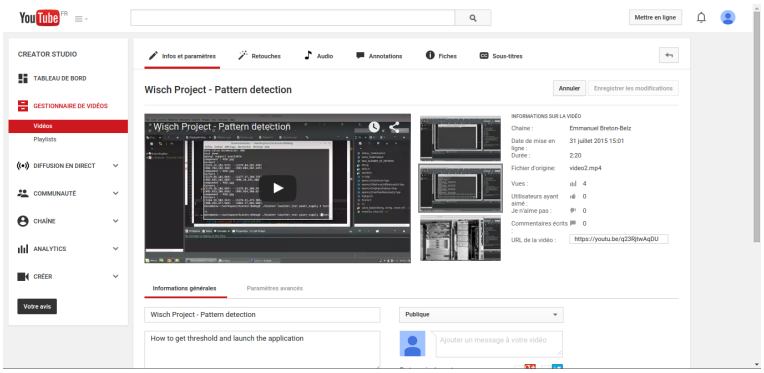

Figure 5.5: Youtube edition page of the video

Chapter 6

conclusion

To conclued, I am really happy of my internship. Even if the first project went wrong, I learned that sometimes we have to stop before we waste our time. I appreciated the location and the communication between Pierluigi and me. I felt followed and supported during the most important phases of the project. That was both motivating and reassuring.

I learned a lot about compilation, that will be really useful for the next libraries I will install. OpenCV gave me a good idea of how far we can go with image analysis. It gave me ideas for future projects and I proved myself that I am capable of realize a project. I also improved my English and learn a little bit of Italian.

Of course, being in a foreign country was an exceptional experience. Every day was a new opportunity to make discoveries and see how other people live. Now I know that in the middle of an unknown country, I will have the better relfexes and I can find my way.

In addition to all of that, I am now sure that a double diploma really is my idea of the last year of my studies. I had doubts about it, but now I am almost sure that I want to move in different countries in my future work.

For all these reasons I want to thank again all the persons who help me to realize this internship.

Bibliography

Chapter 7

Annexes

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

- **Code of the DMatch Structure :**

```
struct DMatch
{
    DMatch() : queryIdx(-1), trainIdx(-1), imgIdx(-1),
               distance(std::numeric_limits<float>::max()) {}
    DMatch( int _queryIdx, int _trainIdx, float _distance ) :
            queryIdx(_queryIdx), trainIdx(_trainIdx), imgIdx(-1),
            distance(_distance) {}
    DMatch( int _queryIdx, int _trainIdx, int _imgIdx, float _distance ) :
            queryIdx(_queryIdx), trainIdx(_trainIdx), imgIdx(_imgIdx),
            distance(_distance) {}

    int queryIdx; // query descriptor index
    int trainIdx; // train descriptor index
    int imgIdx;   // train image index

    float distance;

    // less is better
    bool operator<( const DMatch &m ) const;
};
```

List of Figures

Summary

My internship took place in Fisciano, a little city at 15km of Salerno and 35km South of Napoli in Italia. I was working in the MIVIA laboratory: Macchine Intelligenti per il riconoscimento di Video, Immagini e Media. Which means intelligent machines for video, images and media recognition. They currently work for multinational companies to startups, they are specialized in computer vision and automatism. That is why they often work with drones and robots.

My first project consisted in gathering information about the Motorola HC1 Helmet which is used by companies to provide hands free communication on the field. Then I had to replace the Windows CE operating system for a Linux. I tried a lot of things, but without the references of the components and the skills to erase Windows CE. I could not achieve this task.

After three weeks working on the helmet, I finally switched to another project. A project that fitted more with my skills because I had to implement an image analysis algorithm based on OpenCV. I had to extract patterns on images. The main difficulty of this project was that the pattern could appear several times on the image and they were several patterns to be found each time. This project is a step of a bigger project that pretend to simplify the maintenance manuals by simply scan the machine you want to repair. Because the manuals are often big and a little part of the information concern your machine. The result of the analysis should pop a manual in pdf gathering only the information you need.

I learned about OpenCV and tested code before I really worked on the project. The first version of the algorithm was not satisfying the need so I restarted from zero. Using what I have already done, I easily surpassed the previous algorithm with a new matching system and a strictness in organization. I made a github: `https://github.com/manumanmax/hardwarePatternDetection` and I often reported with my superior Pierluigi.

I finally succeed in my task of suppressing the points representing each pattern when I found one, threshold the matcher and gathering the in a structure. Then Pierluigi asked me to create a video to showcase how the algorithm works.

I acquired lots of skills in computer sciences, but also in embedded system, English and I learned a little bit of Italian. At school, I made 3 internships and this one was the most interesting and relevant. I really enjoyed it because both the place and the work were great.