

Training report

Qt Camera Manager



HiST, Trondheim, Norway

Norwegian mentors

Tomas Holt
Grethe Sandstrak

French mentor

Thomas Diette

1st April to 30th June

by Antonin Durey





Thanks

I would like to sincerely thank Tomas Holt and Grethe Sandstrak for guiding Thomas and me throughout the project. They gave us many wise advises and tips to help us to work on the application and to do the best work possible.

Thanks are essential to Jan Nielsen, and more generally HiST about their great welcome. I quickly felt at home in Norway, and my questions always find answers.

I would like to thank Patrick Lebègue and the 'Relations Internationales' IUT's department for presenting us abroad universities to realize our training period.

Finally, I would like to thank my French tutor, Thomas Diette, for following me week after week and coming us in Trondheim to see our project and our job.



Résumé

Durant mon stage, j'ai travaillé dans le laboratoire de HiST sur la détection de points et la visualisation d'images en trois dimensions. Thomas et moi avons été chargé d'améliorer un logiciel fait par les étudiants français venus l'année dernière. Ce logiciel prévoyait, entre autres, la configuration de caméras, et la visualisation d'images provenant de celles-ci.

Le programme était fait en langage C++, avec une librairie graphique nommée Qt. Ne connaissant ni l'un, ni l'autre, j'ai commencé ces trois mois par l'apprentissage du C++ et de Qt. À cet apprentissage est venu s'ajouter celui de la librairie permettant la liaison avec les caméras, nommé FlyCapture, puis celui de la librairie OpenGL.

Une fois le C++ et ces librairies maîtrisées, j'ai pu commencer à améliorer le logiciel. J'ai notamment travaillé sur certains fichiers de configuration pour rendre certaines opérations automatiques. La visualisation de coordonnées en trois dimensions a également été l'un des temps forts puisque j'ai découvert une programmation graphique totalement nouvelle.

Ce stage m'a bien sûr permis d'accroître mes compétences techniques, notamment par l'apprentissage de nouvelles librairies. Il m'a aussi fait découvrir une autre culture, d'autres méthodologies de travail, et m'a permis de perfectionner mon niveau d'anglais.

Abstract

During my training period, I worked at the Hogskolen i Sor-Trondelag laboratory about points detection and 3D images visualization. Thomas and me were made responsible for improving a software made by last year French students. This application dealt with, amongst others, configuring cameras and visualizing images coming from them.

The software was made in language C++, with a graphic library named Qt. I did not know neither the language nor the library, I began these three months by learning C++ and Qt. To this learning, the camera library, named FlyCapture, and the OpenGL library learnings was added to.

Once this job was over, I started improving the program. I especially worked on some configuration files to make some operations become automatic, such as choose the cameras combinations. Three dimensions data visualization was also a great highlight because I discovered a graphic way of programming totally new.

This training period allow me to increase my technical skills, especially through the libraries learning. It also make me discovering a new culture, other work methodologies, and make my English level increase.

Table of contents

Introduction	7
I) From Norway to the project	8
1.1) Norway	8
1.1.1) Why Norway	8
1.1.2) Short introduction about Norway	8
1.2) Hogskolen i Sor-Trondelag	9
1.2.1) Norwegian school system	9
1.2.2) HiST	9
1.2.3) Vizlab	10
1.3) The project	11
1.3.1) The existing project	11
1.3.2) The different improvements	11
II) Taking in hand, configuration and visualization	12
2.1) Getting used to the project	12
2.1.1) C++	12
2.1.2) Qt	13
2.1.3) FlyCapture	14
2.1.4) The existing documentation	14
2.1.5) Reading, understand and improving the code	14
2.2) Adding graphical components	15
2.2.1) The menu bar	15
2.2.2) The project tree	15
2.3) Calibration_summary.dat	16
2.3.1) Summary of all calibration files	16
2.3.2) Navigate into the file	16
2.3.3) Write, order and hide calibration groups	17
2.3.4) The table view	18
2.4) Socket.dat	19
2.4.1) The table view	19
2.4.2) The table view	19
2.5) 3D programming	21
2.5.1) OpenGL and QtGL	21
2.5.2) Points and shapes	21
2.5.3) Scale, translation and rotation	22
2.6) Other tasks	23
2.6.1) Grupper images	23
2.6.2) Documentation	23



III) Assessments	24
3.1) Human assessment.....	24
3.1.1) Working in team.....	24
3.1.2) Receive instructions.....	24
3.2) Technical assessment.....	25
3.2.1) Practising.....	25
3.2.2) New language and new libraries.....	25
Conclusion	26
Appendix	27
Appendix n°1 :Qt Camera Manager proceedings.....	27
Appendix n°2 :Last year interface.....	28
Appendix n°3 :Improvement of the project interface.....	29
Appendix n°4 :Calibration summary file.....	30
Appendix n°5 :Grupper images.....	31
Glossary	32
References	33



Introduction

Second-year student at the Institut Universitaire de Technologie of Lille in computer science, I was brought to carry out a training period in a company computer science service or in a foreign university. I chose the second option, and went in Trondheim, Norway, at Høgskolen i Sør-Trøndelag. I worked in a real time 3 dimensions motion capturing and visualization laboratory, named Vizlab. The goal was to take over the project began by French students who came last year, and improve it with several functionalities.

People working at Vizlab were used to deal with Trackpoint, which is their 3D motion capturing software. Our project anticipated to configure cameras and Trackpoint, run it, and visualize the out images and files. The major problem we faced was how to automate the most tasks while keeping the software the most clearly as possible.

The project for Vizlab persons was very important because it had to make them gain the most time possible. The majority of the functionalities prescribed could already be done using another way, but the major aim was to gather them together, and automate the most of them.

In a first part, I am going to talk about HiST and Vizlab. I will also explain the project I took over, its functionalities, and the tasks I was supposed to realize. Then, I will explain and describe the tasks I realize, beginning from the graphical improvement to the 3D display, going through the files edition. Finally, I will make out the technical and human balance sheet which will show the skills I gain throughout my training period.

First section : From Norway to the project

1.1) Norway

1.1.1) Why Norway

At the end of the DUT¹, we had to realize a three months training period. We could choose between doing it into a company in France, or in an foreign university. I was not hesitant at all because doing it abroad was one of the reasons I went into the IUT² of Lille. I decided to choose a European northern country, and my choice bears on Norway. I already came in Norway once, and I appreciated the country. Their standard of living, their welcome and their culture convinced me to go in this Scandinavian country.

1.1.2) Short introduction about Norway

Norway counts around five million people and is one of the richest country of the world, thanks to its oil and gas reserves. However, these reserves drained, and the question about the 'After-oil' remains.

The computer science domain is very important in Norway. In fact, to answer the 'After-oil' question, several companies saw the day to develop and improve this field in Norway. We can quote :

- *Opera Software*, founded in 1995, which is of course the company which developed the Opera browser. The Opera is also the browser used in Nintendo DS and Wii.
- *FileMail*, specialized in huge files transfer since 2008.
- *Qt Development Frameworks*, see more page 13.

Located at the centre of Norway, Trondheim is the third city in Norway after Oslo, the capital, and Bergen, with a population of around 180 000 inhabitants. There is around 30 000 students in the city, what makes Trondheim the biggest student city in Norway. Most of them come from the *Norges Teknisk-Naturvitenskapelige Universitet*, but some, as me, study at *Høgskolen i Sør-Trøndelag*.



Illustration 1: Nordic map

1 See glossary

2 See glossary

1.2) Hogskolen i Sor-Trondelag

1.2.1) Norwegian school system

The Norwegian primary and secondary school system is quite similar to the French one, except Norwegian students do not have an exam at the end of the secondary school, as we have in France. The higher education in Norway has, among others, six general universities, six specialised universities and about twenty university colleges. Unlike France, nearly all the higher education institutions are up to the State. The higher education is conform with European rules, and provides Bachelor, Master and Doctorate certificate. There is also a two-year certificate named Hogskolekandidat, which can be completed to get a Bachelor degree, provided by university colleges, as HiST.

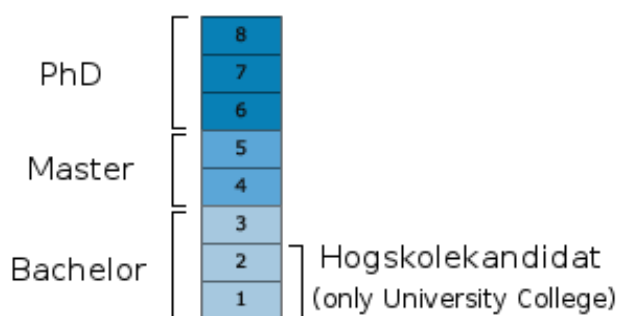


Illustration 2 : Norwegian university system

In France, the higher education is also conform with European rules (except the Bachelor degree is called 'Licence'). However, a number

sizeable of degree are delivered by private school, no matter the study field. In this way, degrees are very different, and sometimes, it is very difficult to evaluate the importance and the quality of degrees.

In France, I was in a Institut Universitaire de Technologie (IUT) in order to be awarded of a Diplôme Universitaire de Technologie (DUT), which is also a university degree. This is a two-years degree which was created to allow students to quickly enter labour market. But the figures show around 80% of the IUT students carry on studies after being awarded. That is what I am going to do, because next year I will incorporate a engineering school.

1.2.2) HiST

HiST is a university college, and the second university in Trondheim after NTNU. It is also the second largest university college in Norway. In fact, there is around 8 000 students. It gets six fields of studying : Health and Social Work, Informatics and E-Learning, Teacher and Interpreter, Technology, Trondheim Business School, and Nursing.

The Informatics and E-Learning domain contains four programmes which all provide a Bachelor degree after three years : Computer Engineering, Network Administration, Information Technology and IT-supported Business Administration.

The IUT I was contains a single programme. We were mainly studying Computer Programming, but also networks, databases, and some generic subject as management, laws (generic and computer science), and the unavoidable mathematics and English.

1.2.3) Vizlab

Inside the Faculty of Informatics and E-learning at HiST, a motion capturing and visualization laboratory has been established. "The project's main goal is to develop an accurate, portable, low-cost system that integrates real-time 3D motion capture and visualization (R3DCV) of human and industrial objects' movements in complex environments¹".

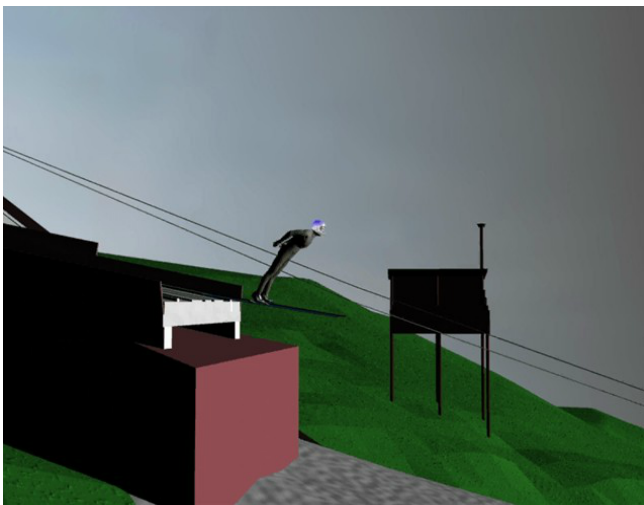


Illustration 3 : Visualization of a ski jumper

Doctorate students collaborated in a project named "Real time markerless motion capturing and visualization of human movement" in 2009.

The aim was to find methods for capturing motion with ordinary video cameras without the use of markers, and use this to control an avatar in real time.

Another project realized by doctorate student was a "Geometric Modeling of Human Depth Perception in a 3D Virtual Environment" project, made the same year. The aim was to work about the depth perception in a virtual environment. The relationship between the different parameters (as brightness, linear perspective...) and perceived depth were studied and compared to a geometrical model based on gaze tracking measurements².

Some students participate in the research projects. In 2003, a project was made about filming ski jumpers from different angles in a ski jump with three synchronized video cameras. Using methods derived from photogrammetry and computer vision can be performed to coordinate accurate 3D points position in the jump skis and the body of the jumper.

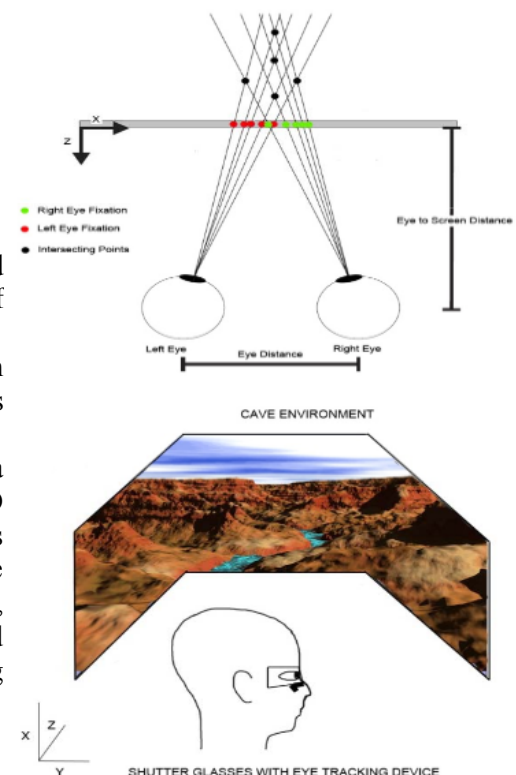


Illustration 4 : Geometric Modeling in a 3D virtual environment

¹ See <http://hist.no/content/24483/R3DCV>

² All taken from http://vizlab.hist.no/documents/2009_11_23_Hoestmote_Short.pdf

1.3) The project

1.3.1) The existing project

We did not start out with nothing. In fact, we took over the last year French students' project, named **Qt Camera Manager**. It is a project to control and set up numerous cameras which run on both Windows and Linux . The program will help to configure the cameras in order to do real-time three-dimensional video motion tracking. The project was made in language C++, with the libraries Qt and FlyCapture¹.

1.3.2) The different improvements

The first day we met our internship mentor, Tomas Holt, we were given a list of some tasks we should realize. These tasks include, among others :

- be able to run the project on both platforms Windows and Linux, 32 and 64 bits
- read and complete the existing documentation
- show coordinates files (it is file produced by the 3D tracking software - Trackpoint) in several views, as text view, table view and 3D view.
- make possible to edit configuration files used by Trackpoint, using several editing level
- choose markers on images used in input
- know if it would be possible to port the application to mobile platforms

In fact, launching **Qt Camera Manager**, we should be able to configure the different cameras with the different files furnished by Trackpoint, launch Trackpoint, and visualize the coordinate files in output².

We were explained the main quality of the software had to be **clearness**. Even if the software would be used by computer science people, its use had to be controllable in a few minutes, and very simple to understand.

¹ See next page for C++, pages 13 and 14 for Qt and FlyCapture

² See [Appendix 1 : Qt Camera Manager proceedings](#) page 27

Second section : Taking in hand, configuration and visualization

2.1) Getting used to the project

2.1.1) C++

The project we took over was made in C++. It is an object-oriented language developed from C in the 80s and standardized in 1998. It includes basic functionalities of C, as the standard library, and basic functionalities of object-oriented languages as classes, polymorphism or inheritance¹. Today, it is the third language the most used behind C and Java.

I did not studied this language during my DUT, but I learnt Java, which is also an object-oriented language. In this way, I quickly learnt C++ through the relationships the two languages have.

However, I had some troubles about the namespace concept, partly because Java does not have this concept. In fact, the C compiler does not allow two entities as attributes, functions or variables having the same name in a same part. To solve this problem, namespaces were added. Namespaces are contexts, as identifiers for the compiler to know where a function, class or keyword comes from. For example, the C++ standard library is included into the `std` namespace. That is why we need to write `std::cout` and not only `cout` to print something. The other possible is to explicit at the beginning of the file that the namespace is used, writing `using namespace std`.

Unlike Java, C++ gets header files to store function signatures, class members and some other informations. Nevertheless, I was used to manipulate header files since my beginnings in language C last year, and this also was not a difficult step.

```
1  #ifndef HEADER_FILE_H
2  #define HEADER_FILE_H
3
4  /* This is a header file ! */
5
6  class Header {
7  public:
8      Header();
9      ~Header();
10
11  protected:
12      void incrementCpt(int addingValue);
13      bool cptIsPositive();
14  private:
15      int cpt;
16
17  };
18
19  #endif // HEADER_FILE_H
20
```

Illustration 5: header.h, an example of a C++ header file

¹ See glossary

2.1.2) Qt



Illustration 6 : Qt logo

Qt is a cross-platform framework¹ made by *Qt Development Frameworks*, previously named *TrollTech*, which is a Norway-based software company. It provides graphical components in C++ and also some others languages, and graphical classes. Qt is particularly known to be the library on which the KDE graphical environment, one of the most used in the Linux desktop environments, is based.

Qt also furnishes others methods and classes, as `QThread` or `QDir` for the code to be cross-platform. In fact, this point is very important for the software to be portable. The functions I was used to use to program with in C where Unix functions, so I needed to learn the Qt equivalents for the software to work on Linux and on Windows.

I learnt graphical programming with Java months ago, and I rediscovered similar graphical components. Nevertheless, each language has his distinctive features, and I tried to go all over the functionalities and the components to discover how use at best Qt.

One of the most longest mechanism to learn was the signal/slot mechanism. In Java, I was used to program with interfaces as `MouseListener` or `ActionListener`, and reimplement methods with the desired code to make event-driven programming². I also programmed using the Model-View-Controller to separate classes and better organize the projects. With Qt, the best way to do event-driven programming is the signal/slot mechanism.

The aim is having signals, as flags, to trigger methods called slots. The connection between signals and slots is made by the method :

```
QObject::connect(objectSender, SIGNAL(nameOfTheSignalMethod()),
                 objectReceiver, SLOT(nameOfTheSlotMethod()));
```

Signals are mainly already written, but it is also possible to create some. In that case, they should be called by the keyword `emit`. Otherwise, they are automatically called, and can provide parameters to the slots. The latter are functions, consequently they contain code lines, receive signal parameters and can be overwrote by daughter classes. By using many times `QObject::connect`, is it possible to link one signal to several slots, several signals to one slot, or both. This is very useful to maximize the project efficiency.

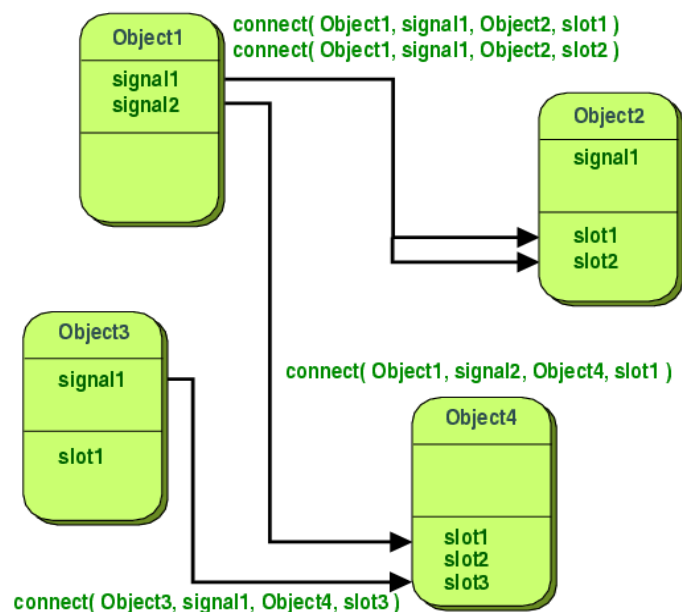


Illustration 7 : Signal - slot concept

¹ See glossary

² See glossary

2.1.3) FlyCapture

FlyCapture is a Software Development Kit created by *Point Grey Research*, providing a camera control library and an image acquisition software named FlyCap2. The FlyCapture installation also configures the IEEE 1394 bus, which is the bus for connecting high-flow peripheral devices, as cameras.



Illustration 8 : Point Grey Research logo

The library furnishes classes for camera representation named *Camera*, camera informations named *CameraInfo*, camera properties named *CameraProperty*, and some classes to record video files or to save image files. The main class used in the project is, of course, the *Camera* class. It provides some vital functions as *Connect()*, *StartCapture()*, *StopCapture()*... Even if the work with FlyCapture was mostly done, I had to master the library to understand the existing code.

2.1.4) The existing documentation

At the beginning, I read the existing documentation to understand how the project was made, and what can be done with it. Quickly, the documentation proved to be unsatisfying. In fact, things explained were too few, and not detailed enough : some Linux commands were false or obsolete.

With Thomas, I wrote again the User Guide and Technical Guide including last year and this year's work¹.

2.1.5) Reading, understand and improving the code

Before writing anything, I tested the software, removed and put again some code parts to see and understand what they did. This was not a step really interesting, but necessary before coding².

Quickly, I detected some improvements easy to program and very useful. The first one was the auto detection of the cameras. I thought it would be useful to do it automatically. So, I made it, using a *QThread*, which was a very good occasion to work again about it. I knew threads³ because I already used it at the IUT, but it was really interesting to work with it in a real situation.

You can see on Illustration opposite, the *ThreadDetectCamera*, which is an internal class of *MainWindow*.

```
/* Internal class to detect cameras */
class ThreadDetectCamera : public QThread {
public:
    ThreadDetectCamera(MainWindow *w) : QThread() {window=w;}
protected:
    void run(){
        window->startCameraDetection();
    }
private:
    MainWindow *window;
};
```

Illustration 9: MainWindow internal class : ThreadDetectCamera

I also made the properties update automatic, using the same thread method.

¹ See 2.6.2) page 23 about the documentation

² See Appendix n°2 : Last year interface page 28

³ See glossary

2.2) Adding graphical components

2.2.1) The menu bar

```
saveConfigFile = new QAction(tr("Save Config File"), this);
saveConfigFile->setShortcut(QKeySequence(Qt::CTRL + Qt::Key_S));

file->addAction(newProject);
file->addAction(loadProject);
file->addSeparator();
file->addAction(loadConfigFile);
file->addAction(saveConfigFile);
```

Illustration 10: part of menubar.cpp

During the test step, I was worn out to always look about the right button. I decided to add a menu bar, with shortcuts to perform actions without using the mouse. Even if this task was not asked by our internship mentors, I thought it could be very useful, and they quickly approved my initiative. I was used with menu bar in Java, and this was quite easy

to understand with Qt. In fact, the menu bar is a `QMenuBar`, which contains `QMenu`. In their turn, the `QMenu` contain `QAction`. The shortcuts are the texts, as `Ctrl+S`, which mean you have to press both `Ctrl` and `S` to perform the associated action.

2.2.2) The project tree

The main task we were told to realize was opening files, which supposed be able to localise files. It could have been more logical to think first about a file list, just showing them one after the others. But the solution was not satisfying. In fact, we had to see in which folder a file was, not only because it was easier to open it, but also because the Trackpoint run depended on many paths in the Trackpoint folder.

Consequently, a tree creation was advised, always to keep the software most clearly as possible. On the project opposite, it is very easy to look for, check and open files and folders.

Creating a tree was also an excellent occasion to rework recursion¹, a notion I already studied, in similar scenarios at the IUT.

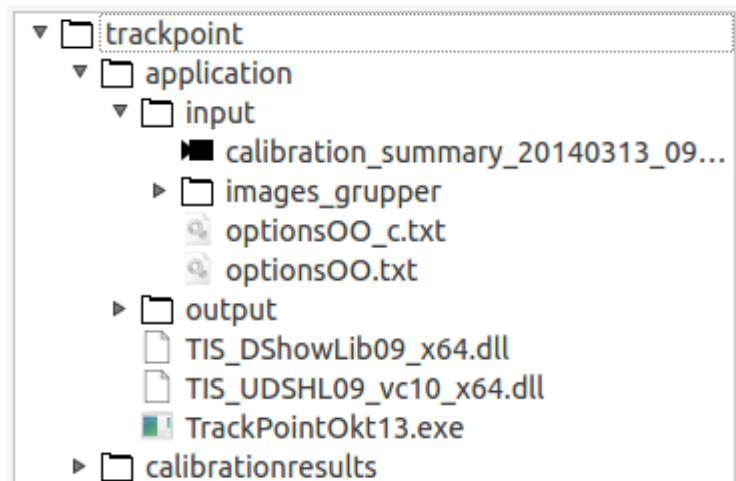


Illustration 11: project tree

¹ See glossary

2.3) Calibration_summary.dat

2.3.1) Summary of all calibration files

The calibration_summary file is the result of the first run of Trackpoint¹. As explained in the [Appendix n°4 : calibrations_summary file](#), it is a summary, but it also contains all the files for all the combinations. Consequently, the file is quite huge, and very long and difficult to read. The first step was to navigate quickly into the file.

2.3.2) Navigate into the file

Even if a scrollbar was obvious to see all the files, it was not quick enough. A person looking for a accurate file could spend many time looking for a file. This is absolutely not the goal of the software, its aim being to gain time. So, I added an action accessible by a left click. The goal is very simple : doing this action on a file summary line, for example *1_2_3: NO CONVERGENCE - NEW IMAGES NEEDED*, will bring you at the beginning of the 1_2_3 combination file.

Going down to find a file is a good idea, but it would be also nice to get up. This has also been done, and with a right click, you can go to the top of the file easily.

These actions, and many others, are made thanks to a very useful class, a QTextCursor. It represents the cursor in a text or line edit field, and can be moved thanks to the function QTextEdit::moveCursor(QTextCursor::MoveOperation, QTextCursor::MoveMode). The MoveOperation field is an enum. The values can be :

- QTextCursor::Start	Moving the cursor at the top of the QTextEdit,
- QTextCursor::StartOfLine	Moving the cursor at the start of the current line,
- QTextCursor::Up	Moving the cursor one line up,
- QTextCursor::Right	Moving the cursor one character to the right.
- ...	

The MoveMode is also an enum which can be either MoveAnchor or KeepAnchor. Opposite to MoveAnchor, KeepAnchor will select the text while moving the QTextCursor. This was very useful to select and remove text.

The four lines below are the main of the 'Go to file' part.

```
/* While it is not the right line, move 'Down' */
do{
    fileContain->moveCursor(QTextCursor::Down);
    lig = fileContain->textCursor().blockNumber();

} while(!lines.at(lig).contains("calibration_comb_"+key+".dat"));
```

Illustration 12: the lines to go down into the calibration_summary file

¹ See [Appendix n°1 : Qt Camera Manager proceedings](#) page 27 for the trackpoint run

2.3.3) Write, order and hide calibration groups

Once moving through the file was completed, the next task was to write into the file, change lines, and especially the first line, which is essential to the Trackpoint second run. Allow the file edition was something easy to do, but it was not enough. In fact, it was very dangerous. Even if only the first is read for the second run, if the user writes without knowing it, or do a slip, all the file, and the second run could be unusable.

To avoid these problems, I was asked to add some actions to minimize the human actions on the file. This included options to choose the combinations for the second run, sort them, and hide some special combinations.

As shown in the [Appendix n°4 : calibrations_summary file](#) page 30, there is 2 sorts of combination lines : the unavailable lines, with 'NO CONVERGENCE', and the others. But, after reflection, I noticed there were others unavailable lines. In fact, because combination work with each other, if one is unavailable, all the combination which work with are unavailable too.

Take an example to better understand : it is a calibration with 6 cameras, so you will need 2 combinations for the second run. If the combination 0_1_2 is unavailable, the reversed combination, which is 3_4_5 is unavailable too.

This report involved many calculations. Worst, with 9 cameras, it was impossible to make strictly calculations. Even if I was not able to realise calculations for more than 6 cameras, I understood why our software was needed : sometimes, some calculations and options are almost impossible to forecast, without trying all the possibilities.

Once the unavailable lines were detected, we should be able to select a line, and not a unavailable one. In fact, the selected lines will be used to rewrite the first line at the top of the file. If an unavailable line would be selected, the Trackpoint second run would not work.

By selecting a combination, some other became useless. In you select the combination 1_2_5, all the others combinations with a 1, 2 or 5 became useless. This second item also involved many calculations.

Finally, I was asked to hide unavailable and useless lines, to show only the main one.

On the file opposite, the unavailable lines are underlined, the select line is in bold (it is the 1_2_5 line), and the useless line are thin. the only line which is not affected is the 0_3_4 combination line, which is the reversed combination of 1_2_5. In the source file, this information is stored with a enum type, as the *Illustration 14* next page shows.

```

1_2_3: NO CONVERGENCE - NEW IMAGES NEEDED
1_3_5: NO CONVERGENCE - NEW IMAGES NEEDED
2_3_5: NO CONVERGENCE - NEW IMAGES NEEDED
3_4_5: NO CONVERGENCE - NEW IMAGES NEEDED
1_2_5: 0.1432. Mean: 0.65, max: 0.88. No of frames used: 55
1_4_5: 0.1725. Mean: 0.63, max: 0.86. No of frames used: 55
1_2_4: 0.1829. Mean: 0.56, max: 0.71. No of frames used: 55
2_4_5: 0.1829. Mean: 0.56, max: 0.72. No of frames used: 55
2_3_4: 0.1859. Mean: 0.57, max: 0.72. No of frames used: 55
0_1_5: 0.1982. Mean: 0.76, max: 0.99. No of frames used: 55
0_2_3: 0.2003. Mean: 0.66, max: 0.84. No of frames used: 55
0_1_2: 0.2143. Mean: 1.21, max: 1.65. No of frames used: 55
0_2_5: 0.2178. Mean: 0.8, max: 1.06. No of frames used: 55
0_3_4: 0.227. Mean: 0.68, max: 0.81. No of frames used: 55
0_4_5: 0.2437. Mean: 0.71, max: 0.88. No of frames used: 55
0_1_4: 0.2453. Mean: 0.72, max: 0.89. No of frames used: 55
0_2_4: 0.2546. Mean: 0.74, max: 0.9. No of frames used: 55
1_3_4: 0.612. Mean: 2.29, max: 3.09. No of frames used: 55
0_1_3: 0.6877. Mean: 2.32, max: 2.93. No of frames used: 55
0_3_5: 0.7349. Mean: 2.41, max: 2.99. No of frames used: 55

```

Illustration 13: camera combinations, written differently to distinguish their status

```

/* This is the state of each camera combination
* 0 : 'NO CONVERGENCE...' or reversed combination
* 1 : Hide because 'NO CONVERGENCE...' or reversed
* 2 : Useless because a combination with one or more cameras in common is selected
* 3 : Hide because useless
* 4 : Normal state, without anything special
* 5 : Selected combination. To be a combination during the record */
enum Calibration{
    Failed=0,
    HideByFail=1,
    Useless=2,
    HideByCalcul=3,
    Normal=4,
    Selected=5
};

```

Illustration 14: enum Calibration to keep in the software each combination status. HideByCalcul should be called HideByUseless.

2.3.4) The table view

The *Illustration 15* below shows the kind of informations we can read in each combination summary. The fact was, with the simple view, the informations were not clearly legible. The idea was to create another view of these files, with informations sorted, and displayed in a table. The *Illustration 16* below shows the result of the table view creation. By comparing the two views, we can easily conclude the table view is much more ergonomic. The user switches between the text and the table view with a menu, given with a right click.

Camno 0. Serial no. 10000
XO: 1550.54 YO: -2332.08 ZO: 1508.39
AL: 0.684638 BE: 0.898275 KA: 0.841765

C: 1279.48 C std.dev.: 8.1341 XH: -31.521 XH std.dev.: 45.1389
YH: -8.657 YH std.dev.: 13.2987 AF: -0.00097 AF std.dev.: 0.001446
ORT: -0.001668 ORT std.dev.: 0.001529
F1: 9.87e-008 F1 std.dev.: 3.58e-008 F2: 8.61e-014 F2 std.dev.: 1.04e-013

Illustration 15: Text view

Camno 0 Serial n° 10000	XO:	1550.54	AL:	0.684638	C:	1279.48
	YO:	-2332.08	BE:	0.898275	C std.dev.:	8.1341
	ZO:	1508.39	KA:	0.841765	XH:	-31.521
					XH std.dev.:	45.1389
Camno 1 Serial n° 10001	XO:	-185.09	AL:	-0.5526	C:	1289.68
	YO:	-2628.77	BE:	1.05218	C std.dev.:	4.1526

Illustration 16: Table View

2.4) Socket.dat

2.4.1) The text view

The socket.dat file is the file where all the 3D coordinates in output are, for every point, at each time. The coordinates are written like this, for a simple line :

Point n°1 x Point n°1 y Point n°1 z Point n°2 x...

Each line corresponds to a time. For example, if you recorded with 9 points during 200 times, you will have 5 400 numbers in a file. Even if they seem to be ordered, you can see on the *Illustration 17* below because there is not enough space on the screen, a moment for the record can be shown in two lines (or more). This is totally unreadable if you want to know which coordinate belongs to which point, which axis and which time. That is why we decided to add a another table view.

```
-0.185917 -0.164009 -0.00831997      0.192266  549.984  -0.052614  69.9583   532.534   193.009
749.726   -0.219481  0.127076  817.574   538.891   246.729
-0.195781 -0.170445 -0.00364489      0.197877  549.96   -0.0270923  88.1552   579.037   167.187
749.727   -0.222685  0.132068  830.003   546.326   271.398
-0.168379 -0.181439 0.00403711      0.231661  549.969  -0.0372513 100.068   571.355   154.261
749.706   -0.217593  0.117403  835.204   523.349   293.536
-0.189889 -0.181277 0.00414504      0.218464  549.997  -0.053432  102.045   496.655   157.975
749.702   -0.217691  0.117093  832.947   448.91    318.863
```

Illustration 17: socket.dat text view, not ergonomic at all

2.4.2) The table view

The first step creating the table view was correcting this line problem, and well align the coordinates. To be sure lines would not split, I put a scrollbar for the user to move left and right to see the whole lines. The coordinates were then put by pack of 3, which represent a point from a camera (x, y and z axis). The result is shown on the *Illustration 18* below.

-0.185917	-0.164009	-0.00831997	0.192266	549.984	-0.052614	69.9583
-0.195781	-0.170445	-0.00364489	0.197877	549.96	-0.0270923	88.1552
-0.168379	-0.181439	0.00403711	0.231661	549.969	-0.0372513	100.068

Illustration 18: First step : clearly separate the coordinates from the different cameras

The second step concerned the time. By scrolling down, the user could be quickly lost the time notion, and it is essential for him to know at what time the coordinates he is looking for were taken. To answer this problem, I added two time axis, on each side of the main table. Scrolling up/down or using the scrollbar will make the 3 widgets move, to keep an perfect alignment. The result is shown on the *Illustration 19* below.

24	-0.161794	-0.180453	-0.00879492	0.22196	549.965	-0.0416608	4.72809	-251.4	24
25	-0.201768	-0.15111	-0.0285821	0.184981	550.016	-0.052248	-112.466	-160.9	25
26	-0.188861	-0.134061	-0.0213069	0.184599	550.031	-0.0470968	-262.123	44.61	26

Illustration 19: Second step : the time axis



The separation between the different points was not enough for our mentors, and I was asked to put different background colours to have a better differentiation. The result is shown on the *Illustration 20* below.

5	-0.181062	-0.157315	-0.00181953	0	0	0	113.531	276.3	5
6	-0.173389	-0.176439	0.00309683	-0.151782	550.091	0.25347	82.0758	154.6	6
7	-0.216234	-0.174035	-0.0087975	-1.2099	549.327	1.53497	69.1292	14.45	7

Illustration 20: Third step : the background colours

Even if for my mentors did not asked me more, I decided to add a help pop up when the mouse is moved over a coordinate. This help pop up gives the user all the informations he needs : the point, the time, the axis and its value. You can see the help pop up opposite.

Point n°0
z : -0.00364489
t : 1

*Illustration 21:
help pop up*

Because sometimes, there would have many points, my mentors asked me to be able to hide points. This could be used, for example, to hide the points used for the scale, or some others useless points.

At the end, this view was very interesting thanks to its organisation. But to have a real idea of what it represents, I had to test the 3D view, and so the 3D programming.

2.5) 3D programming

2.5.1) OpenGL and QtGL



Illustration 22: OpenGL logo

After talking about the 3D view, my mentors talked me about OpenGL to program in 3D. I did not what it was, so I started a new research work.

OpenGL is a API for rendering 2D and 3D vector graphics. The first problem I was faced to was how to make OpenGL and Qt work simultaneously. After some research I found what I was looking for. In fact, Qt has a module called QtGL. This module offers "classes that make it easy to use OpenGL in Qt applications"¹.

QtGL module provides few classes, but I mainly used only 2 of them : `QGLWidget` and `QGLFunction`. The `QGLWidget` class inherits from the `QWidget` class, and, we can use the functions furnished by the `QGLFunction` class inside.

2.5.2) Points and shapes

After creating a `QGLWidget` in the project, I began to make some basic test to learn how it works. As soon as I wanted to draw something, I needed to call `glBegin()` and `glEnd()`. These 2 functions are the delimiters of the drawing functions. `glBegin()` takes a single parameter, which is a macro² which defines the type of drawing. It can be `GL_POINTS`, `GL_LINES`, `GL_TRIANGLES`...

Between these functions, you call only few functions. The main I used was `glVertex3f()` which is a function to draw a point. It takes 3 parameters which are of course the x, y and z coordinates of the point. Up and opposite, you have example of how draw a point, and how draw a line.

There are more useful functions, for example `glPointSize()`, `glLineWidth()`, which take both an integer as parameter, which is in pixel the size/width of the graphical component. I also often used `glColor3f()` to change the brush colour.

This was a very nice way in to learn the 3D programming. In fact, months ago, we made at the IUT a small software to draw in 2D. I rediscover some ways of doing, some similar and some totally different.

```
/* Drawing points at the 'coordinatesShown' time */
for(int i=0;i<(pointsDatas[coordinatesShown]).size();i=i+3){
    glBegin(GL_POINTS);
    glVertex3f(GLfloat(pointsDatas[coordinatesShown][i]),
               GLfloat(pointsDatas[coordinatesShown][i+2]),
               GLfloat(pointsDatas[coordinatesShown][i+1]));
    glEnd();
}
```

Illustration 23: drawing the points at the corresponding time

```
/* Drawing axis lines
 * X axis */
glBegin(GL_LINES);
glVertex3f(0, 0, 0);
glVertex3f(1000, 0, 0);
glEnd();
```

Illustration 24: drawing line example : x axis

¹ <http://qt-project.org/doc/qt-4.8/qtopengl.html>

² See glossary

2.5.3) Scale, translation and rotation

I also work with scale method, which was necessary to show the images properly. Scaling the painting area included know the minimal and maximal value for each axis. I calculated this, and scaled the painting area with the coordinates found. You can see the function used to scale the screen opposite. The `minMax` array contains, in the order, the minimal value for the X, Y and Z axis, then the maximal value for the X, Y and Z axis.

```
/* Scaling the painting area */
glScalef(1/(minMax[3]-minMax[0]),
         1/(minMax[4]-minMax[1]),
         1/(minMax[5]-minMax[2]));
```

Illustration 25: scaling

I did not used the translation, because it was not really necessary but also because all of the mouse buttons were already used, and I cannot find a easy solution for the user. In fact, I thought if the scaling and the rotation were good, the user would not have reason to move the painting area. However, I tried it during my test stage, and the function I used was `glTranslatef()` with 3 parameters, meaning the x, y and z translation.

The thing the most difficult for the 3D programming, and maybe for the whole training period, was the rotation. it was a totally new notion because in 2D programming, you do not have to rotate the painting area. The rotation is provided with the function `glRotatef()`.

Initially, I created sliders, going from 0 through 360 to represent the degree of rotation. But when the user was moving one of the sliders, several things very strange were happening, and I spent hours to understand why. Finally, I discovered I did not understand well how it works: the current angle rotation is not keep in mind, and each time the `glRotatef()` function is called, it makes the painting area rotate with the parameter given. For example, if you called `glRotatef(270, 1, 0, 0)`, you rotate the screen of 270 degrees through the x axis. If you called it again with `glRotatef(271, 1, 0, 0)`, you will not rotate it of 1, but of 271 ! Consequently, I had to change some things in the software. I removed the sliders because it was to difficult and restrictive to keep them, and I kept in the software the previous values.

The second problem I was faced was how rotate the screen without sliders. I choose to press the 'X', 'Y' or 'Z' key, and use the wheel mouse to rotate through clockwise and anti-clockwise.

Finally, one last problem remained. each time the painting area was painted again, the function `glRotatef()` was called. But if the repaint was not called by a rotation update, the rotation function did have to be called. I also kept it in mind to complete this task. The *Illustration 26* opposite show the final result. `xRot`, `yRot` and `zRot` are calculated previously when `mouseMoveEvent` is called.

```
/* Rotation
 * If the last force update was owed by a rotation */
if(rotation){
    glRotatef(xRot, 1.0, 0.0, 0.0);
    glRotatef(yRot, 0.0, 1.0, 0.0);
    glRotatef(zRot, 0.0, 0.0, 1.0);
}
```

Illustration 26: painting area rotation

2.6) Other tasks

2.6.1) Grupper images

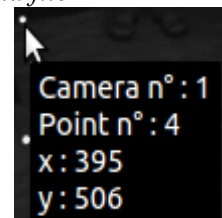
The grupper images are the other way of having images in input for the second run of Trackpoint¹². Because we can see the points on it, I was asked to display the point number, the cameras which took it, and the coordinates values. Contrary to the socket file, these coordinates were in 2D, and not in 3D. In fact, it was the coordinate on the image. These coordinates were stored into some files in the output folder. Because a grupper image corresponds to a time, I had to open the file corresponding to the time of the grupper image file. Down, you can see on the *Illustration 27* the few lines needed to have this corresponding name, and to open the file.

```
/* Having the number of the image (to load the right data file number) */
QString underscoreString= name.split("_").at(1);
QString number = underscoreString.split(".").at(0);

/* Removing the first 0 because the data file has only 4 numbers and not 5 as the image file */
number=number.mid(1, number.size());
QString relativeImagePath = relativePathToDatas + "/" + number + ".dat";
QFile myFile(projectPath + "/" + relativeImagePath);
if (!myFile.open(QIODevice::ReadOnly | QIODevice::Text)){
    cout << "Fail to open the file : "<< QString(projectPath + "/" + relativeImagePath).toUtf8(
    return;
}
```

Illustration 27: Having the grupper image time number, and opening the corresponding data file

Once opening the file, the next step was, once a clicked was detected, to calculate the distance from this point to every point at that time, find the closest, see if the distance was not high, and display the tool tip. This was an unsuspected way to work again about mathematics notions, as Pythagoras' theorem to calculate the distances. The result tool tip is shown on the *Illustration 28* opposite.



*Illustration 28:
help pop up*

2.6.2) Documentation

As I explained in the 2.1.4 part, the existing documentation was not satisfying, and we were asked to complete it. The first step was document the code. Now, every function is described, and the most difficult lines are explained. Even if it added lines, we thought it was essential for the potential people working on this project after us. The second part and third was to write the 'User Guide', and the 'Technical Guide'.

The 'User Guide' is a document explaining each functionality of our software, with which human interaction possibility. This was mainly for the laboratory teachers who will use our software. Even if it was for people knowing computer science, we had to be very conscientious and not forget anything because they could not guess what we have done. If you want to know more about the Qt Camera Manager functionalities, look into this document.

The 'Technical Guide' is a document which explains how work our software, what are the classes, what do they do... Also intended to the potential people working on this project after us. Look into this document is you want to know more about the classes and the way it is done.

1 See [Appendix 1 : Qt Camera Manager proceedings](#) page 27

2 See [Appendix 5 : Grupper image](#) page 31

Third section : Assessments

3.1) Human assessment

3.1.1) Working in team

Working in team was not a something new. In fact, at the IUT, we were used to work in small groups to make projects. In February this year, we even made a project for 12. For this project, I was not if 12 persons but with only one.

The first week, Thomas and I shared the work in such a way as to have parts well defined. I think our main strength was to have a good organisation into the project. In fact, I never worked with Thomas' part, and he never worked with mine. Of course, if we found a mistake, we told the other, but this was exceptional, and we could work into our own part without being in trouble.

Approximately once a week, we were sharing our files to make sure the project was running on both platforms Windows (Thomas) and Linux (me). When we finished our work, our mentors simply gave us new tasks without impinging on the other's part.

3.1.2) Receive instructions

Contrary to the team work, receiving instructions was something quite new. In our projects at the IUT, we were used to have explanations about the surroundings, but no more. We were mostly free to do whatever we wanted, if it felt into the project category.

During the training period, this was totally different. We were not programming for ourselves, but for users. Consequently, we had to listen very carefully about the instructions we received to do the best work possible.

Something very different useful to notice too, we did not have defined work hours. We were told to work, where we wanted (university or home), when we wanted. The only imperative we had was the weekly meeting with our mentors. It was really nice to work like this because we could work in the atmosphere more relaxed than, for example, in companies.

Finally, it was a excellent occasion to speak English a lot; not only our weekly meetings with our mentors, but also with others abroad students. Even if I passed the TOEIC exam in February with 735, I think these 3 months contribute to improve my English level and feel comfortable with it.



3.2) Technical assessment

3.2.1) Practising

Contrary to the projects at the IUT, I was programming almost all the days, several hours per day. This is really better than programming only 5 or 6 hours per week because at the IUT, at the beginning of the each project session, we had to remind what we had done last time, and the tasks to do. This took time, and it was truly appreciated to not do this every day during the training period.

The several improvements I was in charge to do make me work about several things I learnt at the IUT :

- The system programming¹ with the opening, closing and saving the calibration and socket files.
- The event-driven programming, and the graphical programming, when I added graphical components, and handle user interactions with the software.
- Algorithmic problems, for example looking for a path line in a file.

3.2.2) New language and new libraries

I also learnt things completely new. Firstly, I learnt C++ and the Qt library. Many things were similar with Java, so it was nice to learn comparing things I already knew and things I discovered.

I also learnt 3D programming with OpenGL. Even if the most difficult tasks I was given during the training period were about 3D programming, it was fascinating to learn about it. In fact, 3D programming is not something I could practise at the IUT, and even in companies I do not think it is something widespread.

¹ See glossary



Conclusion

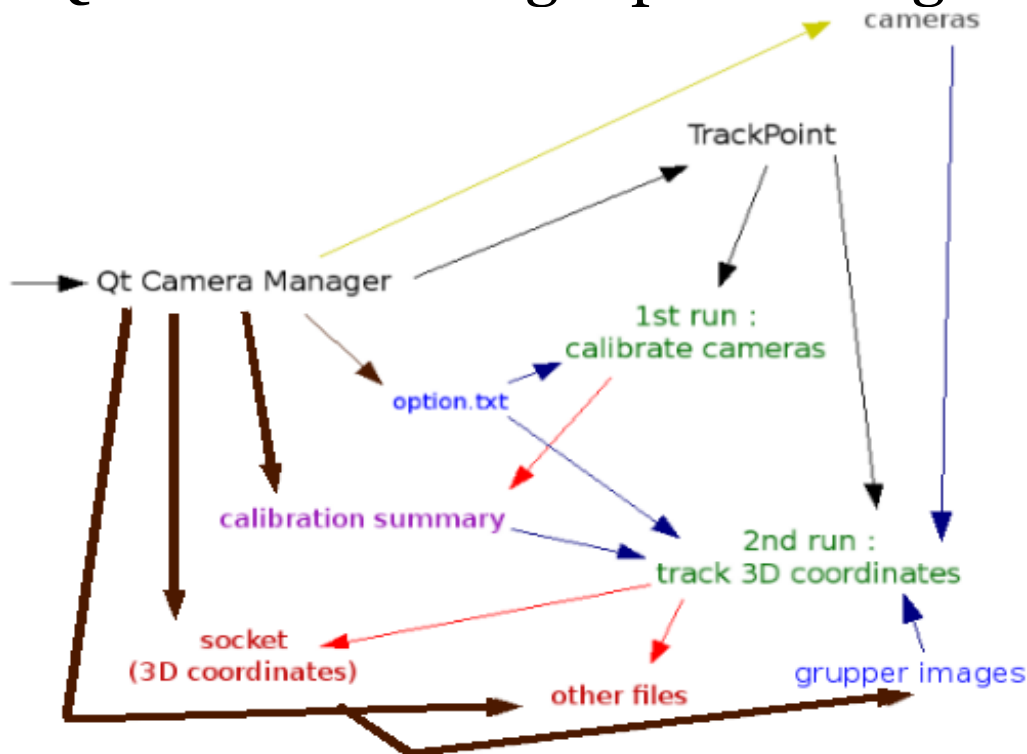
I chose to realize my training period abroad, in Norway, to perfect my English level and discover a new culture and way of life. With Thomas, I took over last year project, which was made by French students. This was a project about cameras configuration and 3D data visualisation.

I mainly worked into configuration and visualisation files. These different assignments allow me to work again into notions I discovered at the IUT, as algorithmic, threads, graphic programming or system programming. I also discovered news things as the C++ language or the Qt and FlyCapture libraries.

The 3D programming was something I totally discovered. This was something not easy to deal with, and I let some tasks unachieved because of their complexity. In fact, if I had more time, I would have worked more on the 3D programming, and especially converting 2D coordinates into 3D coordinates and reverse. I have already spent time on time, but I could not find a satisfactory solution, and I had to move on something else.

All these things gave me new skills in programming. I also gained maturity in project management, whether it be about tasks distribution or about contact between programmers and users.

Appendix n°1 : Qt Camera Manager proceedings¹



blue : input
black : software/run
brown : show/edit
red : output
green : TrackPoint run
grey : hardware
purple : input and output
yellow : check

TrackPoint gets two type of runs :

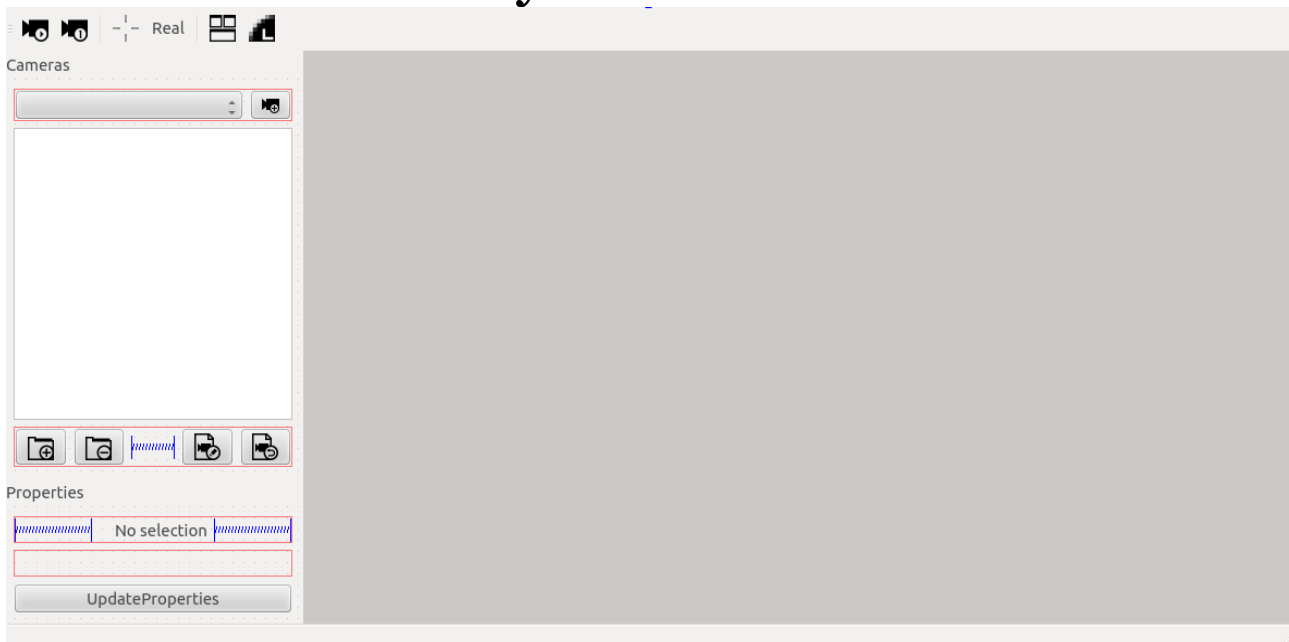
- The first takes the option.txt file in input, calibrate the cameras, and show the calibration results in some files, which the main is calibration_summary.
- The second works with this file and the option.txt file. It takes images for the 3D detection either from the cameras, or from existing images named grupper images. The second run generates many files for each time, a resume of the coordinates which is the socket file and some others.

The goal of Qt Camera Manager is to be able to check cameras and their properties, edit the configuration and calibration files, and show the 3D data in several views. Before, some coordinates views were non existing, and the configuration had to be done manually.

The yellow arrow was done from last year. Thomas worked about the option.txt file. The bold brown arrows show what I have done during the three months.

¹ © to Antonin Durey. Made by myself.

Appendix n°2 : Last year interface¹



Last year project interface

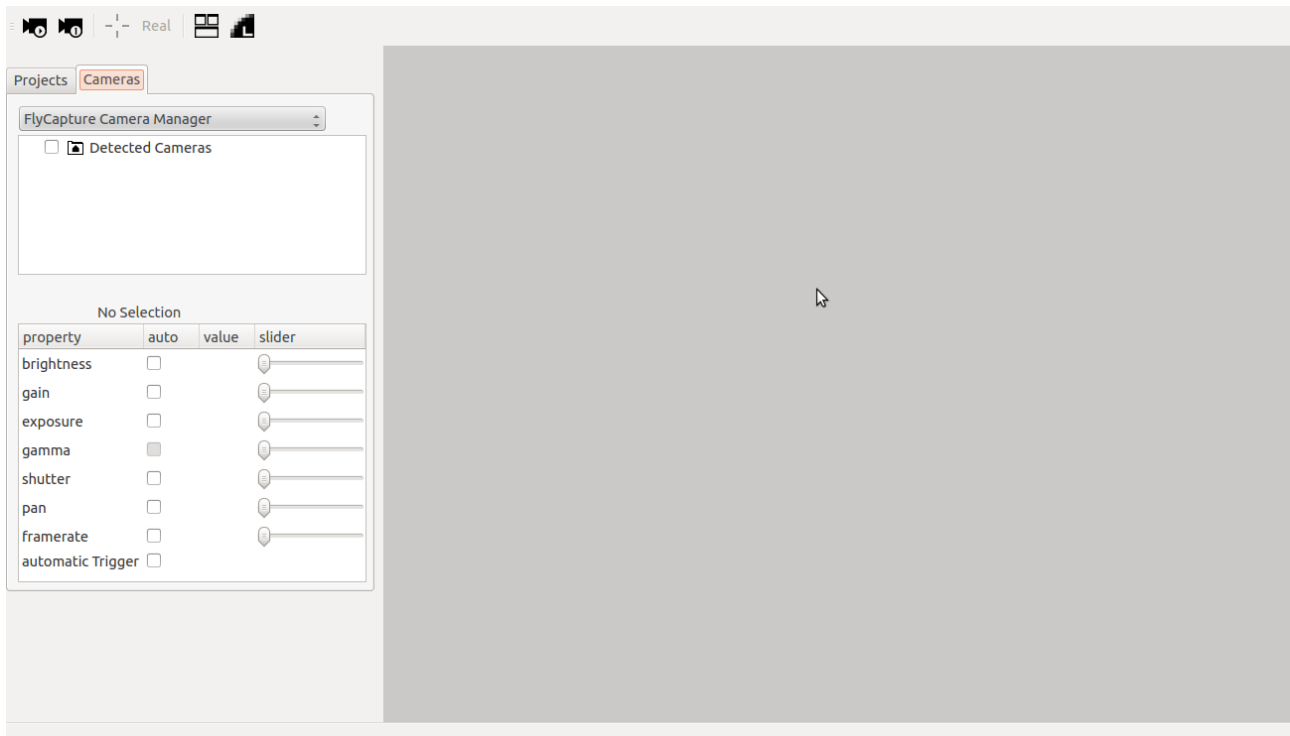
From top to bottom :

- The toolbar, with 6 actions
- The combobox to choose the camera type, followed by the button to update the tree
- The camera tree
- The buttons to change/reset a camera name, add/remove a camera group
- The property title, and the camera name which the properties below are (here, *No selection*)
- The widget summing up the properties and their value. Here, it is just a empty frame, because the image is taken from the *mainwindow.ui*, which is the graphical edition of the software.
- The button to update the properties.

See also [Appendix n°3 : Improvement of the project interface](#) next page.

¹ © picture taken from the software

Appendix n°3 : Improvement of the project interface¹



Improvement on the project interface

The left menu has been split. Now you can see two tabs : the project tab and the camera tab. The button to update the camera tree has been removed : now, it is an automatic update. Same improvement without the 'Update properties' button, it is also automatic.

No shown on the image, a menubar has been added.

See the **User Guide** to learn more about the software functionalities.

¹ © picture taken from the software

Appendix n°4 : Calibration summary file¹

Groups: 1_2_5 0_3_4

1

Image input from file. Directory: input/images_grupper

2

Cam no. 0 has serial no. 10000
Cam no. 1 has serial no. 10001
Cam no. 2 has serial no. 10002
Cam no. 3 has serial no. 10003
Cam no. 4 has serial no. 10004
Cam no. 5 has serial no. 10005

3

Frame and wand type: ORIGINAL, with length: 74990 and coordinates:

pointNo	x	y	z
101	74997	1	1
102	19932	1	1
103	1	1	1
104	1	55012	1

4

SORTED - Camera combinations with S0

6

```
=====
1_2_3: NO CONVERGENCE - NEW IMAGES NEEDED
1_3_5: NO CONVERGENCE - NEW IMAGES NEEDED
2_3_5: NO CONVERGENCE - NEW IMAGES NEEDED
3_4_5: NO CONVERGENCE - NEW IMAGES NEEDED
1_2_5: 0.1432. Mean: 0.65, max: 0.88. No of frames used: 55
1_4_5: 0.1725. Mean: 0.63, max: 0.86. No of frames used: 55
1_2_4: 0.1829. Mean: 0.56, max: 0.71. No of frames used: 55
2_4_5: 0.1829. Mean: 0.56, max: 0.72. No of frames used: 55
2_3_4: 0.1859. Mean: 0.57, max: 0.72. No of frames used: 55
0_1_5: 0.1982. Mean: 0.76, max: 0.99. No of frames used: 55
0_2_3: 0.2003. Mean: 0.66, max: 0.84. No of frames used: 55
0_1_2: 0.2143. Mean: 1.21, max: 1.65. No of frames used: 55
0_2_5: 0.2178. Mean: 0.8, max: 1.06. No of frames used: 55
0_3_4: 0.227. Mean: 0.68, max: 0.81. No of frames used: 55
0_4_5: 0.2437. Mean: 0.71, max: 0.88. No of frames used: 55
0_1_4: 0.2453. Mean: 0.72, max: 0.89. No of frames used: 55
0_2_4: 0.2546. Mean: 0.74, max: 0.9. No of frames used: 55
1_3_4: 0.612. Mean: 2.29, max: 3.09. No of frames used: 55
0_1_3: 0.6877. Mean: 2.32, max: 2.93. No of frames used: 55
0_3_5: 0.7349. Mean: 2.41, max: 2.99. No of frames used: 55
```

5

1 : the *Groups* line : the combination which will be used for the second run are written there (here, 1_2_5 and 0_3_4).

2 : the directory where the grupper images are.

3 : the camera lines, with their serial number.

4 : Some informations useless for Qt Camera Manager.

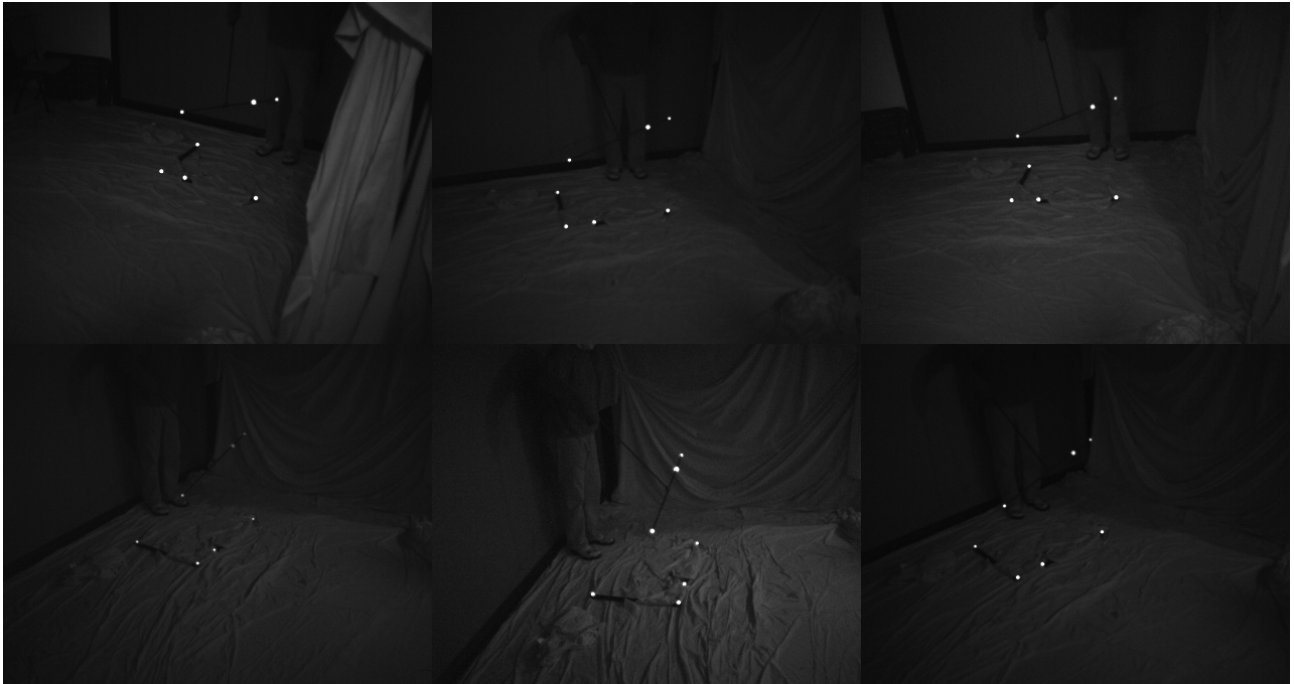
5 : The summary of each combination. The lines are sorted with the parameter written at 6.

Below these lines are all the calibration file, one file per combination. Each combination file is around 40 lines. Counting the combination number opposite, you can imagine the size the calibration_summary can be.

Illustration 29 : begin calibration_summary file

¹ © picture taken from the software

Appendix n°5 : Grupper images¹



Each grupper image is composed of as many images as cameras, one image corresponding to one camera. Because the cameras numbers is always divided by 3, a line is always composed of 3 images. Consequently, the image up comes from a 6 cameras record.

The white points are the points detected by the cameras. There are one grupper image per time.

¹ © to the files produced by Trackpoint



Glossary

class (object-oriented programming) : contains properties representing the object state, and functions representing their behaviour.

DUT (Diplôme Universitaire de Technologie) : French two years degree, provided by IUTs.

framework : components software (such as packets, libraries...) set to create main line software.

event-driven programming : programming concerning the interaction with the users : mouse click, key press...

inheritance (object-oriented programming): notion allowing reusable code through others classes, called descendant classes.

IUT : University element where students prepare a DUT.

macro : rule, defining input sequence format

polymorphism (object-oriented programming) : notion allowing same code to be used in different types, classes, and objects.

recursion : process of repeating items in a self-similar way.

system programming : programming dealing with files, processes...

References

Books :

C++ GUI Programming with Qt 4

by Jasmin Blanchette and Mark Summerfield, with *TrollTech* contribution

Contains basic, intermediate and advanced learnings for Qt. Contains also a "Introduction to C++ for Java and C# Programmers".

Online version : <http://grimaldi.univ-tln.fr/Qt/C++-GUI-Programming-with-Qt-4-1st-ed.pdf>

Websites :

fr.openclassrooms.com

Previously named lesiteduzero.com. French web site about computer science and programming, with tutorials and forum.

hist.no

Official website for HiST.

opengl.org

Official OpenGL website, with documentation, and forum.

ptgrey.com

Official Point Grey Research website, with FlyCapture downloads and documentation.

qt-project.org

Official Qt website, with full documentation, tutorials and forum.

qtcentre.org

Unofficial Qt users forum.

stackoverflow.com

"Question and answer site for professional and enthusiast programmers". My favourite resource during my training period.

vizlab.hist.no

Website about Vizlab, at HiST.