Computer Aided Design - Report

Blanchon Marc - BScV

Summary

Project: Image Processing Tool

Subject: Build a project, using QT IDE with the main topic of image.

Project, Image Processing Tool

Abstract

In this report, we will describe the process of the build of a personal software able to perform processing on images.
The program will be described in a first way for users unable to understand line of codes, by showing interface, explaining the principles without details, and by the “Instructions for use” [High Level Understanding of the Program].
In another way, the report will contain the full explanation of the code, which require a certain understanding of C++ language, QT IDE and general Programing notions [Deep in the Code 1 & 2].

High Level Understanding of the Program

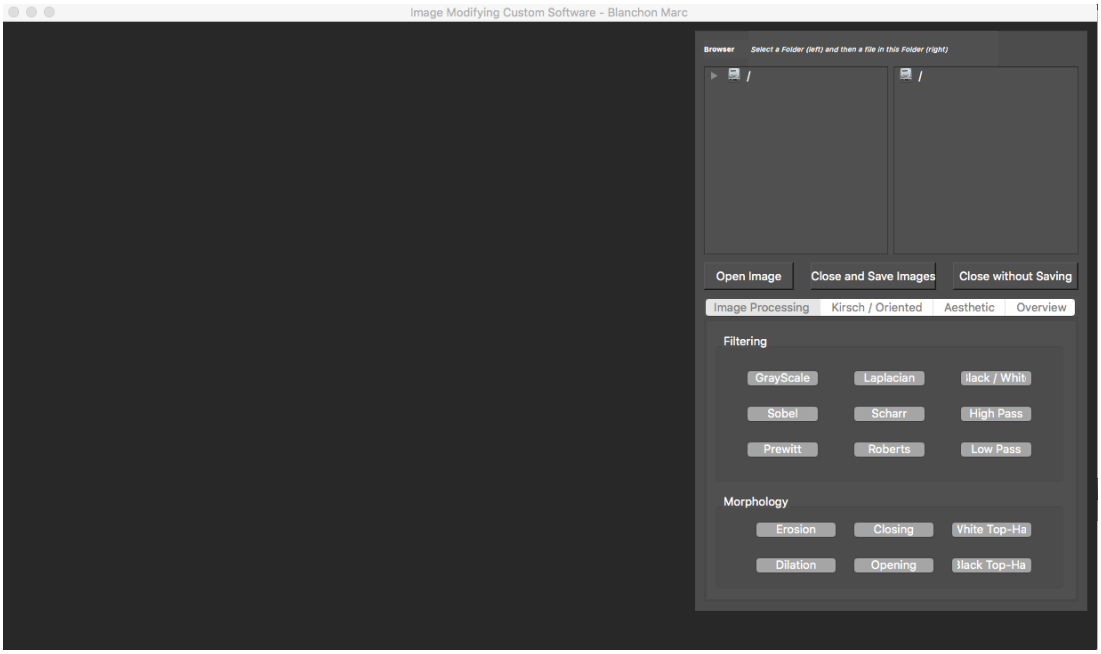
In this section, we will talk about the software which was developed. In fact, we will just recall and look at the goals. And then Explaining with really low detail how the program work for every one which has no affinities with code.

Project, Guidelines and Objectives

During this semester, in computer aided design, we had to build a software and the main subject was the Image.
Before all, I find the Image Processing Tool as entitle and this was in one goal and this is also the subject: showing skills in programing, being able to finish the software before deadline, and having a robust and compatible software “Ready to Use”.

Obviously when we have some kind of homework, this is for showing our skills in this topic. But mainly, I personally add some instructions because we have to be able to build a really strong program, something which is not buggy and almost something that can be commercialized.
This was my objectives. So after the subject choice and these personal instructions, we had a good starting point we kept in mind that the program should be adaptable, understandable for everyone which can look at the code and being as fast as possible.

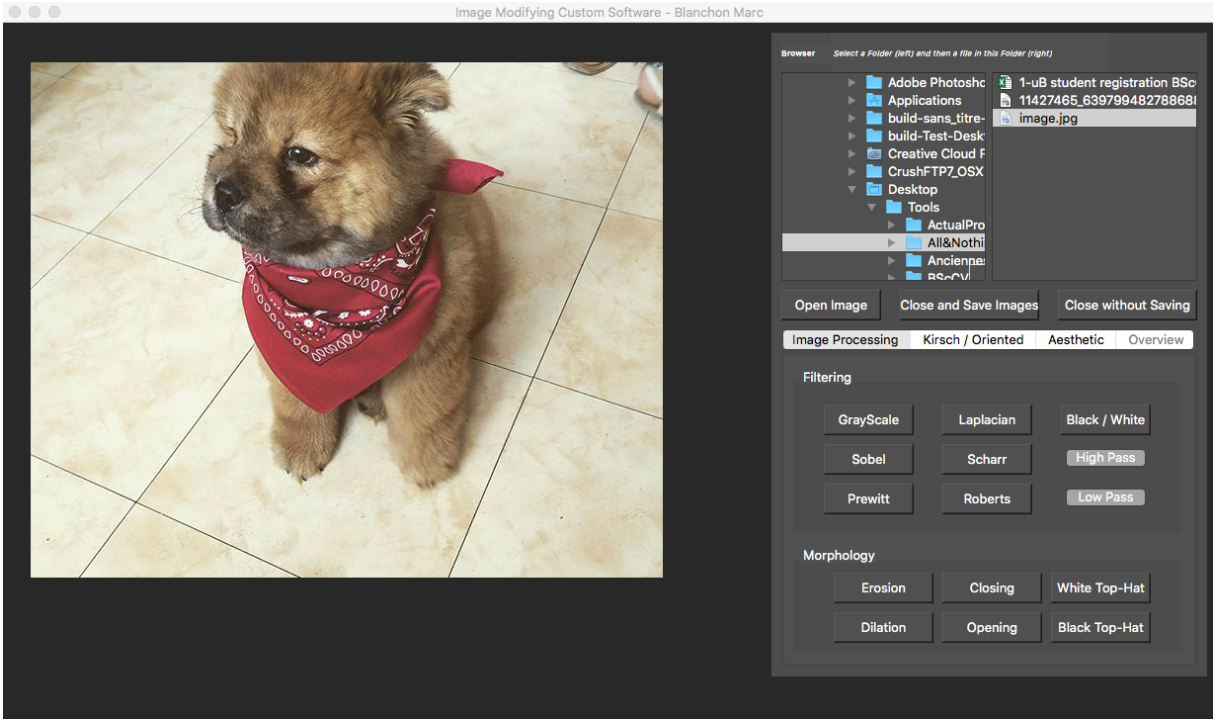
Instructions for use



This is the user interface.
So one of the main guidelines was to have a user-friendly user interface. As we can see this picture was taken on a Mac OSX, so it can have some changes between Windows and Mac. Linux is also a possible distribution for use, and the interface is the same as the one on Windows.

So we can look on the interface. To the right, we can see a big blank space, this is where our image will be displayed.
To display an image, this is simple, only go the file Browser (top right), select a folder on this left panel, then select a file in this folder on the right panel.
Then when you selected the image, you can click on “Open Image”. There is some king of security check, if the file you selected is not an image. The software will advert you and you will have to choose something else.

When the image is opened, you can see changes on the interface.

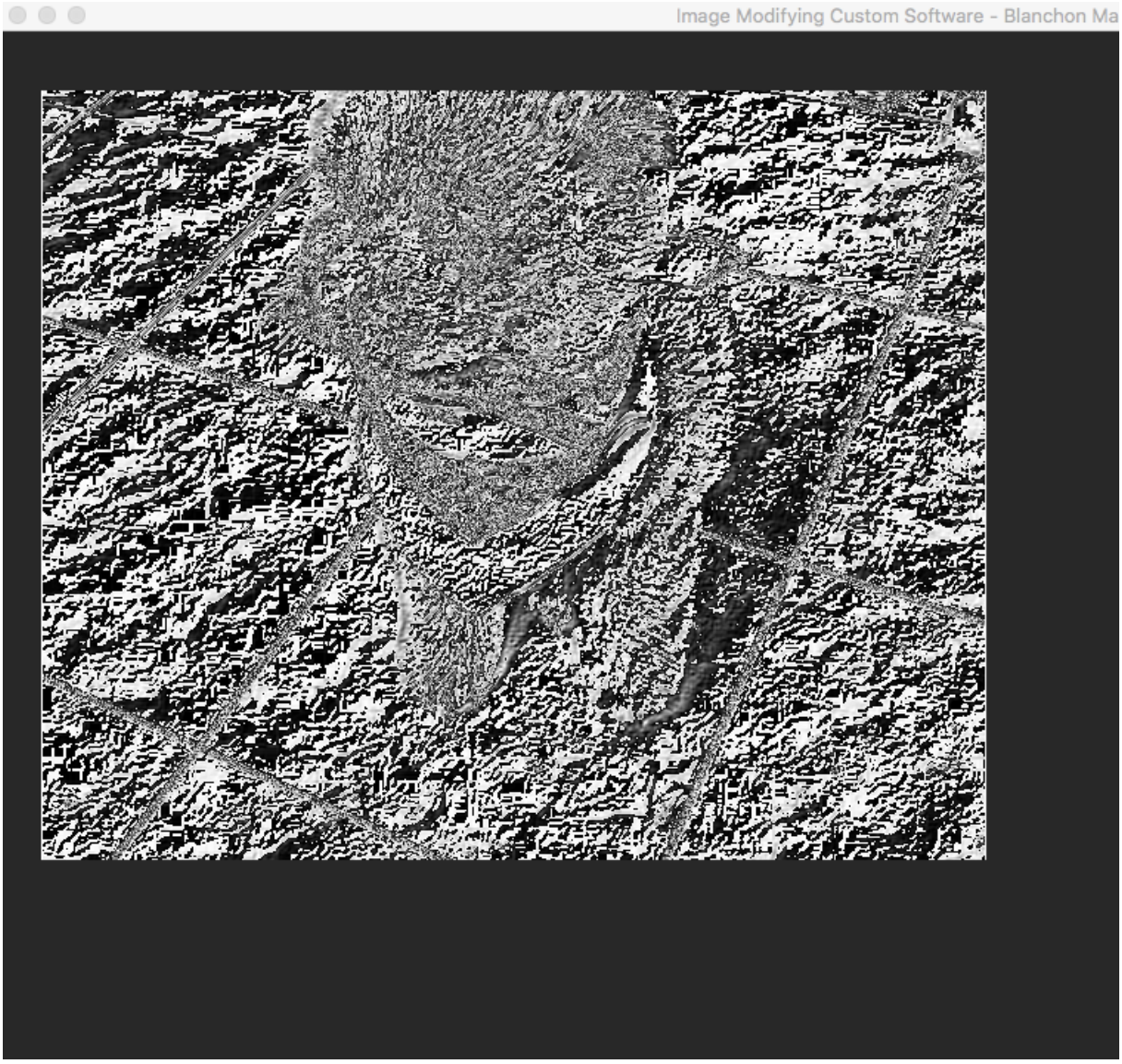


So the image is displayed first, we can also see the directory and selected file on the right. But we also see that our big panel under the File Browser is now open. Before he was locked, it is to avoid many problems, like this you can’t have bad manipulation of the software, you just have to select an image before all.

Now we can have interest on the Buttons Panel. He is cut in 4 tabs each tab corresponds to a kind of processing the image.
We can introduce the light grey buttons / tab this is the “in development” indication.
When you can’t click on something, this is because there is development still in process. So these functionalities won’t be available for moment.

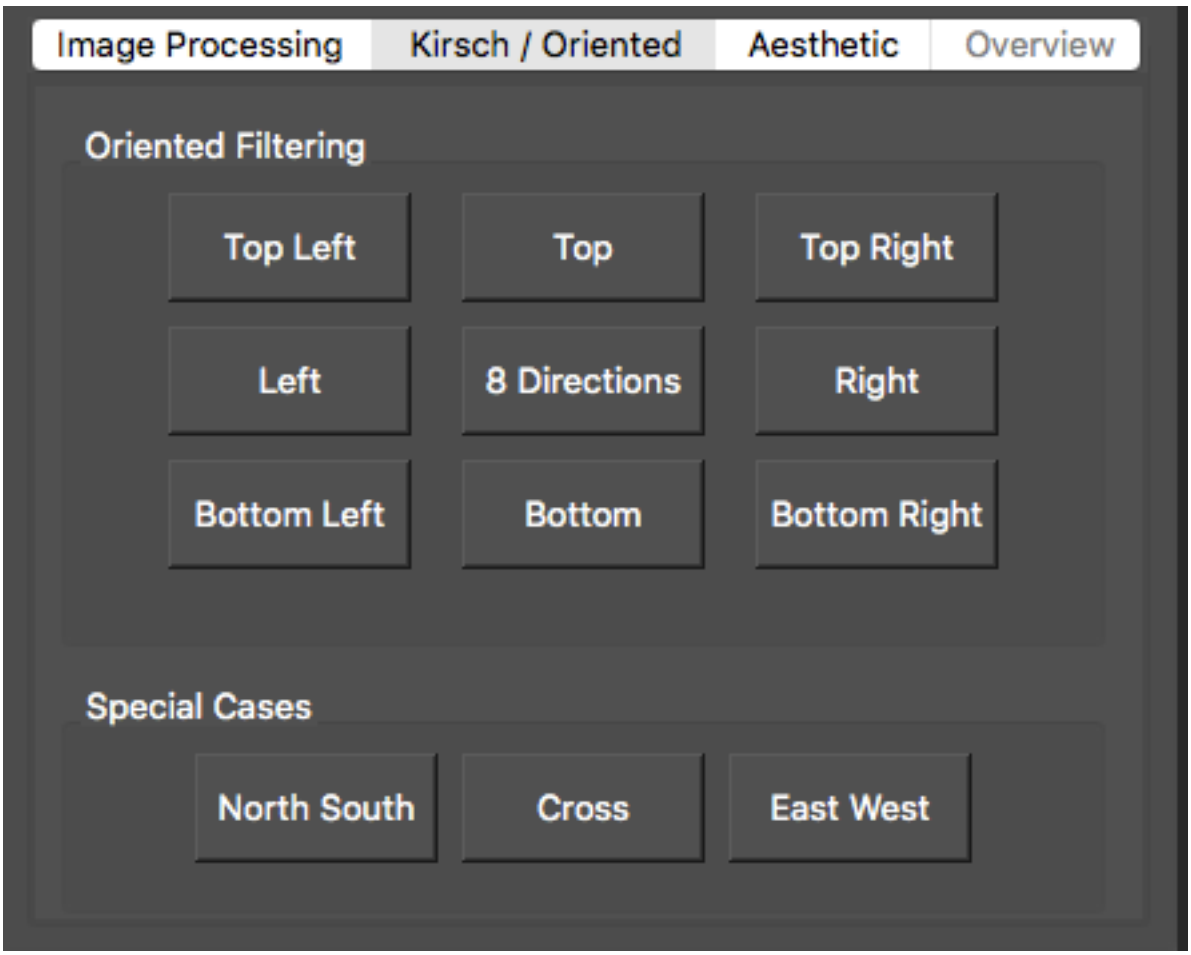
Now we can focus on this “Image Processing Tab”, in fact everything is properly described, if you want to process filter, then you click on the button corresponding to the filter you want. Also, when you want to perform Morphology on image, you can click on each buttons.

Keep in mind that, all time, this is the main image which is computed, there is no cascade, that mean that when you click on “Sobel” the software will compute Sobel on Image. But after, if you click on “Prewitt”, the software will reload the first image, then compute Prewitt.
So this is the meaning of “no cascade”.



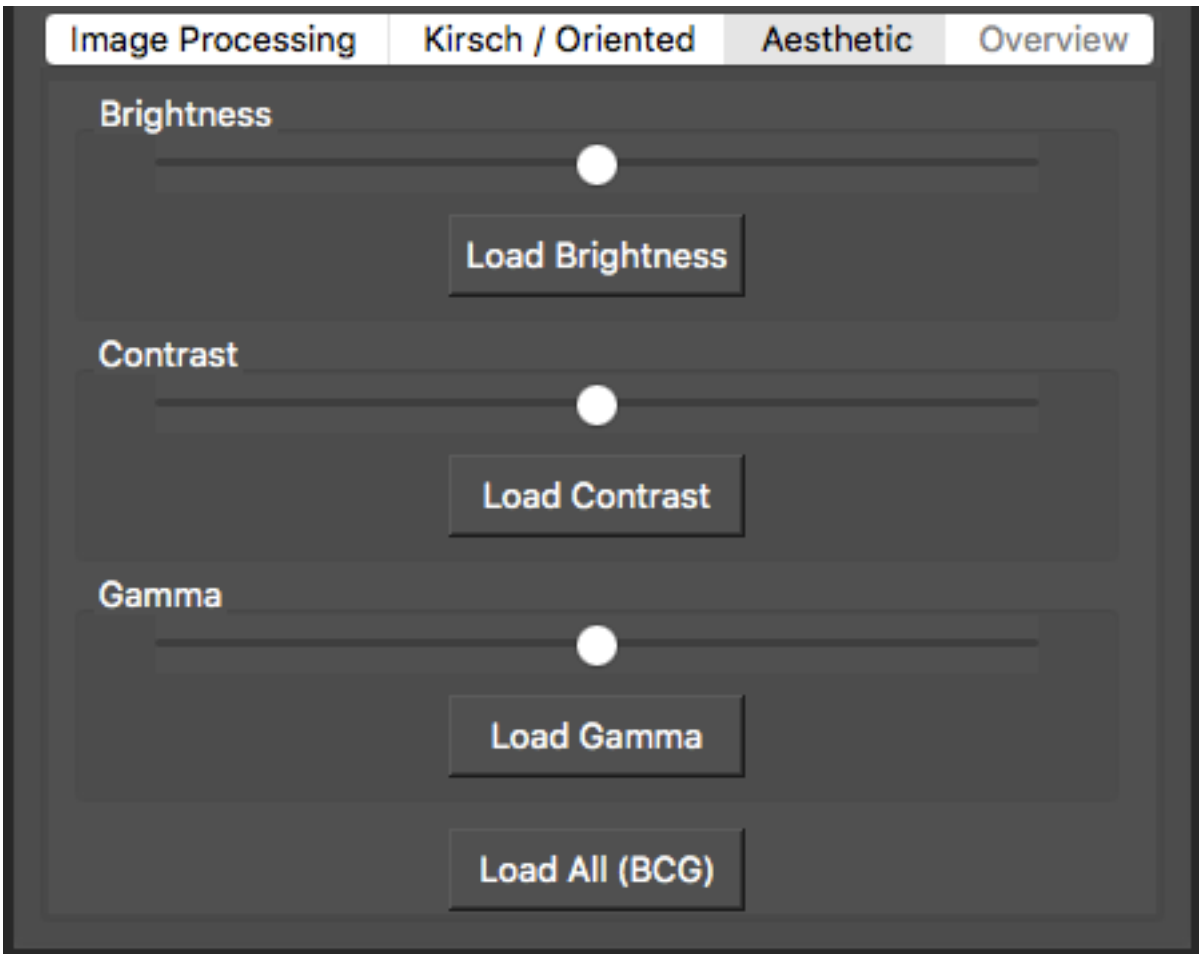
Sobel computed on the example image

We can now go to other tab “Kirsch / Oriented”.



As we developed a user-friendly interface, this is the same principle as before, each things are defined properly.
If we want an oriented filter at 45°, then we just click on “Top Left”, and each filters, are adding or subtracting 45 degrees, just remember that the name of the button mean the orientation, and also the filter. So we can just forget degrees and follow our feelings, then top will preform a filtering from top, to bottom, when bottom right, will perform filtering from bottom right, to top left.

Aesthetic:



Here we can see functionalities that are mainly build in all image tools, so this software is not the exception.

With each bar, you can modify the values of Brightness Contrast and Gamma, each buttons just under the bar, will load only the parameter of the scroll bar.

When you want to perform a full loading of each slider, you just have to click on “Load All”.

To end these instructions for use, we can come back to the closing of the software.
In fact, to understand the closing, we have to know that, this software, each time you click on one button, he will save the image generated in a temporary folder.
This folder is automatically created when your image is, when you click for the first time on a processing button.

So in the mind of the software, you will have backup of all your work. In a folder next to your image. If you already have a folder, and you want to compute things with another image another time, you will open the software, and he will automatically rebuild a new temporary folder, and rename the old one. By this temporary folder named “TempOldX” where X is a number, you will know that the more X is low, the more the work is recent.

To end this, we can explain the two lasts buttons, “Close with Saving” will keep your work on the computer.
And “Close without Saving” will delete the temporary folder.
Obviously each time you click a button like this, you make a decision, and the software will ask if you are sure to perform these actions.

High Level Explaining

So the software is described for a new user just before. But some points are still blur. Here we will talk a little more on details.
The Browser is the access of the user to an image, the folder and files are separated for a sake of visibility.
A button permits us to display image. But behind this, there is a simple process. Each time you click on open, the software is able to check the extension of the file with or without caps, and is able to say if this is a computable file.
This is simple, but really useful to avoid bad manipulations. And this is the same thinking for the tab locking. If your file isn’t loaded or if you file is not an image, then you will not be able to click on functions and like this we can assume that user is safe and the program will not crash for nothing.

We talked before on grey buttons, or disabled buttons. This is the in development functions. We could have just remove all the interfaces corresponding to this, but there is amount of work on these functions, and being disabled don’t mean “is not working”. If buttons are disabled there is a reason, this will not crash the system, but the result is not satisfying so this computation is not useful, so it is disabled.
This is also a way to say that there is improvement incoming on the software, like an update.

Now, we can talk about file managing.
There is a file manager, which is able to create folder, delete folder, create files, saving and so on …
The managing of the file has be decided before the software building. So this was decided that the software file management will act invert as the other software.
Every actions you do, this is saved, every image generated is saved, so instead of saving at the end and having the possibility to lose work, the software is acting like a backup each actions.
At the end we have two options save or not, when you want to save, the program will keep everything. If you choose to close, software will delete every files in temporary folder, and then delete the folder.

To end this section, we mention that the software is not able to cascade, and in fact this is false. You are able to cascade actions, by changing the image in the file Browser. But this cascade function is not developed because I thought that it was not really useful. When you perform a filter on an image, you preform just one, and look at results, and if it is not satisfying, then you perform another filter on base image. This is why cascade is not explicitly build. But this is still possible just using a trick.

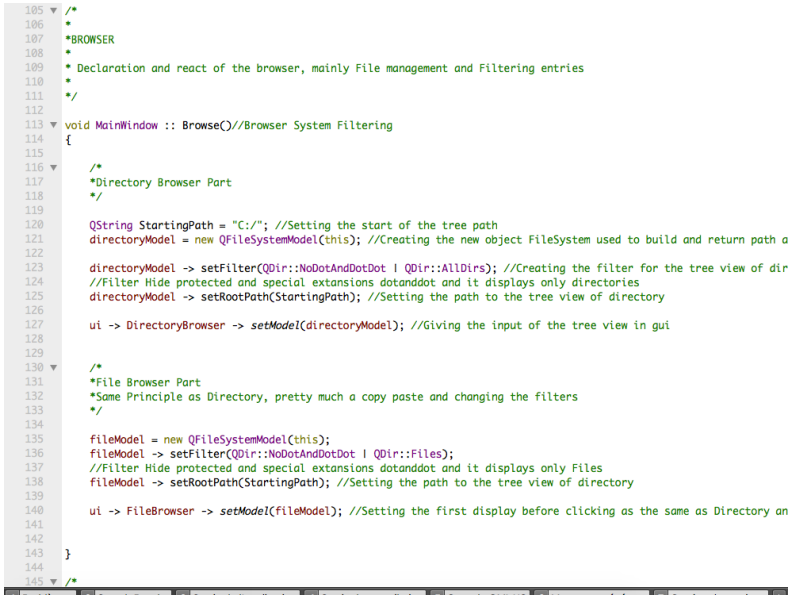
Deep in the Code 1: Generic Functions

In this big section, we will look at how is build the generic function and by generic function this is functions that can be used independently of the program, for another software.
This was one of the main additional objective, being adaptable, so generic functions are created for this.
The purpose of these creations was to mimic the process of professional software engineering, when we create a professional software we have to think before coding and to develop the maximum amount of functions that can be adaptable for a sake of simplicity and software building consciousness.

In a manner of being rigorous, the whole program is commented, indented and all functions and variables are named with logic and locatable names.
If we go in classes (which is not the purpose of this section), comments are quoted, that mean that they have special name, and there is a direct repercussion on the main program, the quotes are also written. Like this we can just copy the comments in classes, and find the same keywords in the main program.

File Browser

The File Browser is the bow on the top right of the interface. It looks easy and smooth but behind this, there is a software which manage this procedure.
This is the function Browse, declared in the MainWindow Header File.



```
/*
*
*BROWSER
*
* Declaration and react of the browser, mainly File management and Filtering entries
*
*/

void MainWindow :: Browse()//Browser System Filtering
{

    /*
    *Directory Browser Part
    */

    QString StartingPath = "C:/"; //Setting the start of the tree path
    directoryModel = new QFileSystemModel(this); //Creating the new object FileSystem used to build and return path a

    directoryModel -> setFilter(QDir::NoDotAndDotDot | QDir::AllDirs); //Creating the filter for the tree view of dir
    //Filter Hide protected and special extansions dotanddot and it displays only directories
    directoryModel -> setRootPath(StartingPath); //Setting the path to the tree view of directory

    ui -> DirectoryBrowser -> setModel(directoryModel); //Giving the input of the tree view in gui


    /*
    *File Browser Part
    *Same Principle as Directory, pretty much a copy paste and changing the filters
    */

    fileModel = new QFileSystemModel(this);
    fileModel -> setFilter(QDir::NoDotAndDotDot | QDir::Files);
    //Filter Hide protected and special extansions dotanddot and it displays only Files
    fileModel -> setRootPath(StartingPath); //Setting the path to the tree view of directory

    ui -> FileBrowser -> setModel(fileModel); //Setting the first display before clicking as the same as Directory an


}
/*
```

As we can see lines are commented and just before a comment which is called before in the report quote, has been written to have a better understanding of someone if the code is analysed or just rebuild by someone else.

On the code, comments are cut but we will explain the process so they are useful in this report.
First we can see that tune function needs no input parameters, because we don't need anything to start a browser.
Then there is declaration of a string, which is only the starting point of the directory browser (on the left part of the box).
Then we just create an object that is able to return files, we set filters to have only folders displayed or special files (this is a special cases), and set the starting point of the browser (here c:/).

Then we repeat the same process for the File Browser but changing filters, to have only files. With only this we have a file browser but he is not working, obviously we need to let him know if we click what he have to do.

```
226 /*
227 *
228 *BROWSER COMPLEMENT
229 *
230 * Only slots functions, permit also to initialize some variables useful in the whole program
231 *
232 */
233
234 void MainWindow::on_DirectoryBrowser_clicked(const QModelIndex &index)//Clicking in the Directory Tree /
235 {
236     QString DirPath = directoryModel -> fileInfo(index).absoluteFilePath();// Getting the absolute path
237     ui -> FileBrowser -> setRootIndex(fileModel -> setRootPath(DirPath)); // Setting the path in FileTre
238 }
239
240 void MainWindow::on_FileBrowser_clicked(const QModelIndex &index)//Clicking in File tree // Output the P
241 {
242     absoluteFilePathSelected = fileModel -> fileInfo(index).absoluteFilePath();
243 }
244
```

The first function is for the Directory Browser, we can see that this is clicked function so we assume that is will be when we click on something. As input we have an index, in fact, this is the index (the number) of the directory that we clicked on.
After this we can see that we influence the File Browser to display the content of the Folder indexed, we input as root a string created before containing the absolute path of the folder that we clicked on.

After this the Browser is working perfectly but we can see that there is a function called when we click on the File Browser.
This is just a variable setter. It permits to the program to have in memory in a variable the path of the file. This will be really useful during the whole program.

Saving Tools

The saving tools are mainly the possibility to the program to save at each time your work. This is cut in many different generic functions. We can first decompose the main function called.

```
155 void MainWindow::SaveWithName(bool First, QString Name, FilteredImage* FI)
156 {
157     if(First)//If this is the first time, then we will create folder !!
158     {
159         QString Check = absoluteFilePathSelected.split('/').last();//Deleting the Name of file in the path
160         Check='/'+Check;
161
162         savingFolderPathSelected=absoluteFilePathSelected.replace(Check,"");//Setting the Folder path
163         FI->TemporaryFolder(savingFolderPathSelected);//Creating Temporary Folder
164
165         absoluteFilePathSelected.append(Check); // Rebuild the filePath
166     }
167
168     FI->TemporarySave(FI->get_savingPath(),ui->DisplayLabel,Name);//Save Image
169
170
171     /*
172      *OverView in Developpement, Unable to Update ui and unable to display images on a second label
173      *
174     QString overviewPath=absoluteFilePathSelected;
175     overviewPath=overviewPath.replace(overviewPath.split('/').last(),"")+"Temp";
176
177     QDir dir1(overviewPath);//Setting the directory we will display all files
178
179     foreach(QString dirFile, dir1.entryList())//for each file in folder //FOREACH1
180     {
181         Overview.insert(overviewPath+dirFile);//insert unique values of images in the set
182     }//ENDOREACH1
183
184     LoadOverview(imageNumber);
185
```

Before starting we can see a big comment, and there is the closing bracket after. This is what is called “in development”, there is a quote DEV written, and all is commented. We can uncomment all, it will not work but this will not make the program crash.

Now, we can look at the function. SaveWithName is a function that takes 3 input parameters. A Boolean, a string and an object of type FilteredImage.
The FilteredImage will be explained in the third big section “Deep in Code 2” so we will just assume that this is an object passed by pointer.
The Boolean is a variable that we will see a lot and lot of time in the program, this is in fact the variable that permits us to know if this is the first time the program is launched.
To explain this, we remember in instruction that is you click on filter the first time, it will create a folder, else it will save your work in this created folder. This is the exact purpose of this variable, knowing if this is or not the first time.
The string name is only the additional name we will add to our saved images.

After this there is a logic condition, if this is first time, then we are creating before the folder. We are just deleting the name of the file in the absolute path by the three first instruction, then we call a function (dependant of the object) to create the temporary folder.
And then we re-update the path.
After this we call a function also dependant of the object which will save the image in the folder.

This is easy to understand but there are two blur things, the functions called by object.
In fact, these functions are a part of the Polymorphism, that mean that, FilteredImage is a derived class of another biggest one, and these functions are derived and redefined in each subclasses.
We call it with the object for one reason, the reaction of the function can be different between two subclasses, so we consider the object. This is in the objective to have an adaptable program. Yes, in this application we use only one subclass, but if we had another (an it could be the case in our application with GrayScaleImage for example), then our program is able to say, you have already polymorphism, just adapt the reacts of these functions in function of your class.

So here this is not really necessary to use this kind of architecture, but this is optimized and we think about the next programmer that can have another idea of the image processing by stocking all different images in different types. This is also what we called before software engineer consciousness.

Deep in the Code 2: Image Functionalities

In this section we will see some advanced notions of C++ such as polymorphism, heritage, template. These notions will not be totally described because this is a report on a project and not explanations on C++.
We will have some call and recall on the previous sections, we will also skip some parts of the code because the report has to be concise and some functions are not really interesting to be explained. This is a choice to present some functions and not others too because this may be give the will to explore the code and find all the tricks and methods used to build the program.

Class and Derived

In this part, we will look at the headers files. As a brief recall, we can remember that a header file is the declaration of all components of the main program. That's why we will see the classes and the use of them.
To avoid redundancy, and to skip the obvious, we will forget the header file of the MainWindow because we are focusing on image in this section and also because all header looks like the same, we can say that in contain all our slots functions, all our variables that are useful in MainWindow and some functions like FileBrowser.

Image Class:

```
class Image
{
public:
    //Constructors
    Image(){}
    Image(QString imagePath) : realPath(imagePath){}
    Image(const Image& I): realPath(I.get_path()){}

    //Destructor
    ~Image(){}

    //Accessors

    QString get_path() const{return realPath;}

    //Mutators

    void set_path(const QString& newPath){ realPath = newPath;}

protected:

    QString realPath; //Path to the image

public:
    //Functions
    virtual void Display(QLabel*); //Defined in the main class and can be used in the derived one


    virtual void TemporaryFolder(QString){} //Force to declare in the derived class //Create Temp Folder
    virtual void DeleteTemporaryFolder(QString){}//Same //Delete Temp Folder
    virtual void TemporarySave(QString, QLabel*, QString){} //Same //Save in temp folder



};
```

Subtitle2

This is our main class which will be derived.
We can see the constructors, destructor, accessor and mutator to respect the standards.

The constructors are divided in 3 separate constructor, this is because we can have multiple declaration of creation of our class. In fact, this is some kind of generic standard to take account of possibilities. We will mainly use the second one that declare the Image with a path and then in the class you can see the instant declaration and initialization of the realPath variable.

Let the report make a little break to talk about names. We can see that there is many different name called, we can discuss about them.
As more as possible, the variables are declared with normal letters and when we add another word, we add a caps on the first letter (like realPath). Functions are declared using capital letters for each new words and the first too. And to end on standards of the code, we see that accessors and mutators are all normal letters, words separated by an underscore. We will see also later that some functions have capital letters and underscore, there is a simple explanation, we declare the function as something and then application and follow this schematic for all functions (example MyFunction_Filter).

Now we can see that there is mutator and accessor, we will not explain them because the program is understandable but we can talk about the interest of these functions.
This is used because our only one variable is protected, this is used to have no access by the other functions or member of the program. So to have access to the variable, we will use mandatorily accessor or mutator.

After this, there is a public keyword and more functions, and these functions are virtual. And here come the Polymorphism but this is a special one. For the first function, the goal was to develop a derived function, so virtual, but also to be accessible from the class Image.
We will skip the code of the Display function but we can explain the concept. We are able to call an object image, that will be able to display without the call of a derived function.
And if needed this function can be overloaded by the derived class. This is the concept used for the first display of the image. This concept is here to differentiate the first image to others.

Then we have three functions, derivable virtual, but they are pure that mean that there is no const=0 behind them. And this is the consequence of the first function Display. We had a problem because we were calling a function in a class that should be pure but wasn't. So the solution is to delete all const=0 and like this we are able to access this function and let her being derivable.
All the other functions will be derived in the derived class.

FilteredImage Class:

```
49  class FilteredImage : public Image
50  {
51  public:
52      //Constructors
53      FilteredImage() : Image(){}
54      FilteredImage(QString imagePath) : Image(imagePath) , savingPath(Image::get_path()) {} //Path to the image savi
55      FilteredImage(const FilteredImage& FI) : Image(FI.get_path()) , savingPath(FI.get_savingPath()) {}
56
57      //Destructor
58      ~FilteredImage(){}
59
60      //Accessors
61
62      QString get_savingPath() const{return savingPath;}
63
64      //Mutators
65
66      void set_savingPath(const QString& newPath){savingPath = newPath;}
67
68  protected:
69
70      QString savingPath;
71
72  public:
73
74      /*
75      *
76      *UTILITY FUNCTIONS
77      *
78      * Poly from Image Class, we can see const and not const
79      * When the const keyword is not used, it permit us to manipulate variables and change values this is why there
80      * While const is here, we assume that there will be no variable changes
81      *
82      */
83
84      void TemporaryFolder(QString);
85      void DeleteTemporaryFolder(QString) const;
86      void TemporarySave(QString, QLabel*, QString);
87
```

```
144     /*
145     *
146     *AESTHETICS FUNCTIONS
147     *
148     * The Names of functions are explicit and says exactly whatr is does
149     *
150     */
151
152     void BrightnessChange(QLabel*, QString, int) const;
153     void ContrastChange(QLabel*, QString, double) const;
154     void GammaChange(QLabel*, QString, double) const;
155
156     void LOAChange(QLabel*, QString, int, double, double) const; // Load 3 sliders values at once
157
158     /*
159     *
160     *USEFUL FUNCTIONS
161     *
162     *Functions used to Wrap the code using this kind of functions to hav less repetitions and more readabi
163     *Functions like this will be adaptable, that mean that, we can just copy paste these functions and thi
164     *These functions have been developped to optimize, quality and adaptability of the program
165     *Some Complex functions that needs more than two array or more cascade stay unchanged to avoid complex
166     *
167     *Functions are templated inside a non template class which is called member templating
168     *It is usefull to deal with multiple types of input, like int and double
169     *
170     */
171
172
173     template <typename T> void ApplyOneFilter(QLabel* , QString , T[9] )const;
174
175
176     template <typename Ta, typename Tb> void ApplyTwoFilters(QLabel* , QString ,Ta[9], Tb[9] )const;
177
178
179 };
180
```

We can see two pictures of the class FilteredImage, this is just a sample of the class, there are many functions between these two pictures but we will focus on what's important and there is already all the main components on these pictures.

The first thing to see is that this is a derived class from Image, but we already say this tons of times.
We have the same principles are before, constructors, destructors, accessors and mutators, and this is exactly the same working way as before.
When we declare our new object of type FilteredImage, then we call the constructor number two that will initiate the path of our image.

We can now look below. We have Utility functions and if we look at the names, this is the derived functions from Image.
We can recall a little on Polymorphism, we can declare functions in a main class, derive this class and makes these functions virtual, to have a special comportment for each derived class. This is the principle. We only have one derived class, it's a fact, but we think about later, when we will continue the program or if another programmer touches this program, we can have many interpretation of the image processing and how to compute these procedures. So it looks useless but it is not, because we need adaptability and strength of our program and this is exactly one of the pillar of these notions in the program.

After this, on the second picture, we see aesthetics functions, we will use on of these functions as example.

```
152        void BrightnessChange(QLabel*, QString, int) const;
```

This function can be called as generic, because all functions have the same architecture.
We are declaring the function, we input a QLabel as pointer that permits us to manipulate the user interface, a string that is the path of the image, and the integer variable is an option, in our function we need integer parameter, but this is an option, in other function it can be double or just nothing. The function is constant because she is not modifying anything.

With this architecture, we can already guess how the program work. 80% of the functions of the program are built like this, and this is generic, there is a maximum amount of useful input that allow us to manipulate the interface and the path.

Now we can look below on the picture and this is a tricky part.
We are using template function to wrap the code and make him adaptable. But the trick is elsewhere, we are using template keyword to template only one or two function in a class that is not template.
And the process is really simple, our program is able to compute filters, but filter can have multiple members, int, double, long, float, so by templating these functions, we are able to take account that the type is not a problem, we are doing the same computation, we just change a type of data and it works. Like this in our program we are able to call only one or two function to do the whole job, and we don't need to convert or compute the data, we input it like values independently of the type and this is the strength of Template !

We are using same architecture for these functions, we can just guess by the name of the functions and the inputs, that are arrays, that it will be our images modifiers.

Computation Functions

In this part, we will study only one function, but we talked about many functions before. So we will describe some functions in the next part, and other will be totally forget because this is a report and we can't and don't have to display all our code. Contrariwise, we have to give the desire to explore the code.

```
368 void FilteredImage::GrayScale(QLabel * label, QString path) const
369 {
370
371     QPixmap DisplayedImage(path);//Pixmap to display
372
373     QImage image = DisplayedImage.toImage();//Converting to Image
374
375     //Variable Init
376     int width,height,gray;
377     QRgb color;
378     width=image.width();
379     height=image.height();
380
381     //Double Loop for Pixels
382     for (int i = 0; i < width; i++)//FOR1
383     {
384         for (int j = 0; j < height; j++)//FOR2
385         {
386         color= image.pixel(i, j);//Getting the parameters of pixel
387         gray = (qGreen(color)+qBlue(color)+qRed(color))/3;//Converting in an average gray level
388         image.setPixel(i, j, qRgb(gray, gray, gray));//Setting the Image Pixel
389         }//ENDFOR2
390     }//ENDFOR1
391
392     DisplayedImage=DisplayedImage.fromImage(image);//Converting to a pixmap
393     label->setPixmap(DisplayedImage);//Displaying on label
394
395 }
```

This function is used to convert a color image to GrayScale. And we can also say that this function is not chosen for nothing. This one of the only functions that are not influenced by the template function. The computation is different and the calculus are special that makes no use for templating this function.

We can see that the code is commented and indented. So we can start to explain.
We declare a Pixmap using our path (the path of the image chosen), and we convert it to a matrix of pixel (exploitable) by using the QImage library.

In a little parenthesis we can say that, we used many libraries from Qt or C++ standards (cmath) and we didn't spend time on this. And there is a reason, we used library only in special and obvious cases, when I were able to build a program that do the work instead of library, I preferred to build the code.

So after this, we will continue with the code and see some variable declarations. This correspond to the width and the height of the image and a QColor type variable that will be useful to store the color of each pixel.

After this we can see a double loop, that will parcour each pixel, and we will extract the color of pixel, do the sum and dividing by 3 to have the average GrayLevel computation for each pixel.

After this we convert our QImage into a Pixmap and we display it in the zone of the interface.

In terms of calculation speed, we are able to deal with big images and being able to display results almost simultaneously.

Small Bonus:

We will display another special function which is not template for one reason, there is a math trick in, and this is interesting to see another function to understand the way of thinking of the programmer.

```
1697 void FilteredImage::BrightnessChange(QLabel* label, QString path, int toAdd)const // BRIGHTNESS CHANGE
1698 {
1699
1700     QPixmap DisplayedImage(path);//Pixmap to display
1701
1702     QImage image = DisplayedImage.toImage();//Converting to Image
1703
1704     //Variable Init
1705     int width,height;
1706     QRgb color;
1707     width=image.width();
1708     height=image.height();
1709
1710     //Double Loop for Pixels
1711     for (int i = 0; i < width; i++)//FOR1
1712     {
1713         for (int j = 0; j < height; j++)//FOR2
1714         {
1715         color= image.pixel(i, j);//Getting the parameters of pixel
1716         int newColor[3]={qRed(color)+toAdd , qGreen(color)+toAdd , qBlue(color)+toAdd};//Create table to verify parameters properly
1717
1718         for(int l = 0 ; l <= 3 ; l++){//Verify parameters //FOR3
1719
1720             if(newColor[l]>255){
1721                 newColor[l]=255;
1722             }else if(newColor[l]<0){
1723                 newColor[l]=0;
1724             }
1725
1726         }//ENDFOR3
1727
1728
1729
1730         image.setPixel(i, j, qRgb(newColor[0], newColor[1], newColor[2]));//Setting the Image Pixel
1731         }//ENDFOR2
1732     }//ENDFOR1
1733
1734     DisplayedImage=DisplayedImage.fromImage(image);//Converting to a pixmap
1735     label->setPixmap(DisplayedImage);//Displaying on label
1736 }
```

This function is used to change the brightness of the image in the aesthetic tab, and we see lot of similarity. In fact, this is the same program, almost a copy paste (and that's why we template some functions), and we can see that there is one more parameter as input so there is changes.
The only change is that we are doing other calculation between two loops, and we are considering the three colors separately. Then we look if the color is not out of boundary (0-255) and then we just output the new image with new colors.

The math trick is that, when we want to add brightness to image, we want her to be closer to the white, and absolute white is value 255 of color, so when we input a number in this function, she is able to makes the image closer to the white.

Before we were talking about template functions and why some are not template. In fact, this is a good example, we can see that there is a special computation between the two loops, and this special calculation is not possible to template, or it is but it is really difficult. So instead of spending time on template special functions, the choice was done to keep functions like this, and template all that can be templated.

Generics Functions

In this last report section, we will talk about generic functions, they are redundant in this report and this is because they are mainly used in all the calculation of the program.

There is many ways to interpret generic, so we will explore them, generic can be interpreted as template, and we have template functions in the program, but it can be interpreted as some kind of standard schematic to build program.

Example of standard schematic:

```
529 void MainWindow::on_LPButton_clicked()
530 {
531     FilteredImage* FI= new FilteredImage(absoluteFilePathSelected);
532
533     if(FirstLaunch){
534
535         SaveWithName(FirstLaunch,"BaseImage", FI);
536         FirstLaunch=false;
537     }
538
539
540     FI->LP_Filter(ui->DisplayLabel,absoluteFilePathSelected);
541     update();
542
543     Images.push_back(FI);
544
545     SaveWithName(FirstLaunch,"LowPass", FI);
546
547 }
548
549 void MainWindow::on_EroButton_clicked()
550 {
551
552     FilteredImage* FI= new FilteredImage(absoluteFilePathSelected);
553
554     if(FirstLaunch){
555
556         SaveWithName(FirstLaunch,"BaseImage", FI);
557         FirstLaunch=false;
558     }
559
560
561     FI->Ero_Morph(ui->DisplayLabel,absoluteFilePathSelected);
562     update();
563
564     Images.push_back(FI);
565
566     SaveWithName(FirstLaunch,"Erosion", FI);
567
568 }
```

Here we have two functions, we can see that they're not commented and there is a reason, the first function using this schematic is commented and then we get rid of them because it is useless to explain multiple time the same thing.

So we look at the functions and we see something else, they are a kind of copy paste. This is true, the only changes is the string called in the last function and the FI->Function… where the function is different.

This is what is called standard schematic in my point of view, we are using multiple time the same schematic because what we crated in the code is that generic that you don't need more, everything is taken in account in low amount of lines and minor changes.
Only two names are changing and the computation is totally different. And this really powerful and adaptable, after this we can also say that the code is lean and clean and this was also a personal objective, having a great, clean and optimized code.

We can explain a little, the function creates an object FilteredImage using the path as input, we look if this is the first time we enter in program (remember we talking about this before), if yes, then save the base image in the folder created and set the Boolean to false like this we will not have multiple created temporary folders.
Then we are just calling the function to do as argument of the class, we update the interface to be sure that the image will be displayed (this is not mandatory it works without), then we add our object in a vector (this vector is not used for moment, he should be used in the Overview section which gives not results I want so this tab is not presented).
And at the end we call exactly the same function, saving, with a new name.

We can also look at the functions that the object owes and this is exactly the same process.

```
424 ▼ void FilteredImage::Roberts_Filter(QLabel * label, QString path) const
425   {
426
427       int ValueTab1[9] = {1,0,0,0,0,0,0,0,-1};//Declare the filter1 [1 0 0 ; 0 0 0 ; 0 0 -1]
428       int ValueTab2[9] = {0,0,1,0,0,0,-1,0,0};//Declare the filter2 [0 0 1 ; 0 0 0 ; -1 0 0]
429       this->ApplyTwoFilters(label,path,ValueTab1,ValueTab2);// Call generic function to apply filters on image
430
431   }
432
433 ▼ void FilteredImage::Laplacian_Filter(QLabel * label, QString path) const
434   {
435       int ValueTab1[9] = {0,1,0,1,-4,1,0,1,0};//Declare the filter [0 1 0 ; 1 -4 1 ; 0 1 0]
436
437       this->ApplyOneFilter(label,path,ValueTab1);// Call generic function to apply filter on image
438
439   }
```

We have the example on two functions, one use 2 filters, the other 1 filter, we declare a new table, and we input it in a template function (cf page 16) independently of the type.
All the functions are the same for the generic case. This is clean and powerful.

Template function:

In this small part, we will display only the smallest template function because this is already a big function and with two filters, it will be so big and in fact we are doing the same thing. Instead of building 1 image from the base image, we build 2 images and we are approximating the sum and it gives our result.

But let's focus on one filter.

```
template <typename T>//Templated to be able to deal with int or double arrays
void FilteredImage::ApplyOneFilter(QLabel* label, QString path, T ValueTab1[9])const //Generic Function to appy one
{//Enter label where display image, path of the main image, and the filter as an array of 9 elements
    QPixmap DisplayedImage(path);//Pixmap to display

    QImage image = DisplayedImage.toImage();//Converting to Image

    //Variable Init
    int width,height;
    QRgb color;
    width=image.width();
    height=image.height();

    QPixmap imageFirstPix(path);
    QImage imageFirst = imageFirstPix.toImage();//Converting to Image

    int sumPixel,count=0;

    //Double Loop for Pixels ignoring contour using filter
    for (int i = 1; i < width-1; i++)//FOR1
    {
        for (int j = 1; j < height-1; j++)//FOR2
        {

            sumPixel=0;
            count=0;

            for (int l = -1; l <= 1; l++)//FOR3 //Double loop intermadiate to see all neighbour
            {
                for (int m = -1; m <= 1; m++)//FOR4
                {

                    color= image.pixel(i+l, j+m);//Getting the parameters of pixel

                    sumPixel+= ValueTab1[count]*color;//Setting the values in function of counter and matrix vector
                    count++;

                }//ENDFOR3
            }//ENDFOR4
             }//ENDFOR4

             imageFirst.setPixel(i, j, qRgb(sumPixel, sumPixel, sumPixel));//Setting the Image Pixel

        }//ENDFOR2
    }//ENDFOR1



    //Double Loop for calculating the approximation
    for (int i = 0; i < width; i++)//FOR1
    {
        for (int j = 0; j < height; j++)//FOR2
        {

            if(i == 0 || j == 0 || i == width-1 || j == height-1){//IF1
                image.setPixel(i,j,qRgb(255, 255, 255));//White contour
            }else{

                color=imageFirst.pixel(i,j);
                image.setPixel(i,j,qRgb(color,color,color));
            }//ENDIF1


        }//ENDFOR2
    }//ENDFOR1



    DisplayedImage=DisplayedImage.fromImage(image);//Converting to a pixmap
    label->setPixmap(DisplayedImage);//Displaying on label
}
```

Because of the picture there is two time the closing bracket for the fourth for.

So we will just explain principle. When we have a filter, we take our image without contour (we ignore the first and last row and first and last column), and we apply our filter to the image.
The filter is giving coefficient, you multiply the value of image by the coefficient corresponding to the position in the mask, and you do the sum of all these multiplications to have the value of the new central pixel.

These explanations are easier with the code, we are ignoring the type, because this is template, and we have a table of 9 cases, which will correspond to a filter 3x3 matrix.
Then as before we are looking at each pixel but this time we are ignoring contour (start and stop condition of for loops).
Then we see another double loop, and this is our matrix parkour. For each pixel, we will look at each neighbour of him, and apply the filter. Then it gives us a sum, and we apply this sum to our new image as color (because this is a color, coefficient * color = color).
After this we have a filtered Image with our array inputted independently of the type.

After this we build the contour in white, and apply pixels on our image and to finish, we display the image.

This is the template function, after this as I explained before, for two filter, we are doing it with two filter, on two different images, we are doing the approximation of the sum and then we display the result.
This is exactly the same process.

After explaining this, we can ask “Why template function when Copy Paste can be used?” and there are many reasons.
Adaptability is the first reason; we will have a more adaptable program.
Clean the program is also a reason, we have all time the same thing, we don’t need to confuse our mind with tons of code when we just can create a generic function that deal with every of our problem.

Results and Improvements

We can talk about results, our program is working and we have good results. But I think we could have better results and smooth results using OpenCV library.
But, I'm happy to be able and to have build this program without using any built-in libraries that can do the work for me.

We can see many artefacts on filtered images and this can be because of lot of things, precision, image quality…
We still can detect contour for the filters of edge detection and this is the main purpose.

As results we can talk also about how the project was going. And to be honest, my time management was really bad, at the start of the project, I had no real motivation and started building some classes only, after some time, I had problems and then I totally stop the work and loose the whole motivation (and this is not an excuse). After some time, I came back with a lot of motivation and really the will of succeed in my work and in this project, and I also had some big ideas like the File Browser and the File Management system. And this gave me motivation.
So in a sake of honesty I prefer say that my time management and project management was really bad, and there is no excuses for this, even my problems should be separated from work / studies and this is one of my defect.

There is still a point that can be explored and this is the Improvements.
We can do many improvements on this project and I'll write a list to store them and maybe to continue the project on my own with these objectives:

- Adaptation on OS, (problems of compatibility with Mac)
- Finish HP / LP Filters and 8 Directions
- Finish / Building the Overview of the Working Folder
- Add Option to change the user interface color
- Add the possibility to cascade filters without File Manipulation in File Browser

Conclusion

This project was really interesting and as I said before to be honest, I didn't get motivation from the subject at the beginning. I think perseverance is the main thing that save me from not being able to have a good project.
By this I'm not saying that the amount of work is not big, I spent a lot of time on my project in a little interval and that is the problem of time management.

To conclude on this report, we can say that we have seen the big lines of the project, passing from real High Level Understanding to a more complex C++ approach, explaining the main ideas and the principle of the program all this by trying to be as concise as possible.

To conclude on the project itself, we can say that as close as possible, the guidelines are respected, we have a user friendly interface, clean and lean code, lot of notion of C++ and some kind of optimized code.
I was / am happy to work on this project and this is finally interesting, because there is lot of improvements possible.
I am still working on the software and after lot of time spent, I still have a lot of interest in this project because there is many things to develop and there are infinite possibilities.