The project

Is developed and prepared by
AZADEH HADADI

Submitted in partial fulfillment of the requirements for the master course of
SOFTWARE ENGINEERING

In Degree Program
MASTER IN COMPUTER VISION

At
UNIVERSITY OF BOURGOGNE FRANCHE-COMTÉ

2019

Abstract: the project is focusing on design, development and testing of a software kit which allows exploring city of Le Creusot which includes finding a path between two points or trajectory planning between two points via an intermediate point, similar to Google Map with some functional requirement. The tools has already been developed and tested from the scratch as an independent and offline tool (independent from GPS or any other smart devices and works without internet connection). All data including, city maps, the coordinate of the place, name of the places, location marker and any other dependencies have been stored locally in the project repository. The project is designed and implemented so that it can be easily understood and the tool can be used with minimum difficulty. To this end, if any algorithm is used in the program a brief explanation was given. Since, it is interesting for everybody to compile and test the program, this part will be described first. After, the report will detail the minimum requirements of the project, which is given during the project briefing session. Then, based on this requirement, tools will be explained. Functions as well classes and graphs will be detail one by one. At the end, a little demo will be presented to give and insight and a guideline on how to use this development kit.
This tool shall be improved to be fully functional in any sense and to satisfy user needs in practice. Thus, improvement never stops and always there is a new aspect to explore and improve. For that reason, some future works will be proposed at the end of the report to be used as a guideline for those eager to improve this tool.

Table of Contents

Introduction

This report is arranged in following sections to give a smooth follow of information to the read:

1. Project definition
2. Compiling and testing
3. Functionality requirement and user expectation
4. Tools and libraries required to implementation
5. Map modality and information visualization
6. Algorithm design
7. Development and implementation
8. Testing and some results
9. Future work and improvement

A brief description of the project was given during the project briefing, which will be used as project definition and primary requirements. This will be restated exactly as given in the first section. The current development will not strictly stick to these requirements and tries to propose a new set of tools to make the implementation of these requirement a bit faster and easier which helps the future developer replicate the development easier for a new city. Section 2 will explain how the code can be compiled on a new computer equipped with Qt IDE and the setting needs before compilation. This section will be present first, because some users are interested to use the software package as a black box or a software module and they are not totally interested in knowing what is going on inside the code. This section will help them to get what they need easy and quickly and if the software kit is not doing what they expect then they can give up readying the rest of the report and follow other solutions and save their time. Functionality requirement will be replicated from project definition in section 3.

OpenCV library will be used in entire this project for visualization. Therefore, in section 3, the compilation of the library and employing its various functions and classes will be detailed for Qt environment. This guideline is very vital for compiling, attaching the library to the project source, setting operating system environment variable to work with QT IDE and visualize the result and the map. Some other tools of map display and map search tool will be presented in this section. Algorithm such as path planning or shortest path between two given points will be discussed here. Loading different map modalities and information visualization will be discussed in section 5. The algorithm design, functions, and classes will be discussed in section 6. In section 7, the software will be implemented and each pierce of code will be detailed. GUI also will be discussed in this section and the functions and variables will be explained. At the end, some future work and improvement will be suggested in section 9.

1 Project definition

The goal of this project is to develop a software to locate (and rate) various buildings, such as schools, university buildings, major offices, hospitals, various shops, cool restaurants, streets, roads, parks, and other interesting points in Le Creusot. The key idea is to develop a software to visualize le Creusot, its streets, some buildings, roads, and parks, in which the user can perform several actions, such as asking for an itinerary between two (or passing through more) points, or ranking (or adding information) to a bar or a restaurant for example. The software should be designed so that a newcomer to Le Creusot should use it to find useful information, such that detail under following section.

- What is the shortest path to walk from the IUT to the Centre Universitaire Condorcet?
- Can you provide an itinerary that is at most 5km long, that passes by a bakery, that starts in Résidence Jean Moulin, and that ends in the Résidence Acacia?
- How can I walk from the Centre Universitaire Condorcet to a grocery store
- I want to visit the Parc de la Verrerie...
- How much time does it take to drive from the train station to Condorcet?
- How many grocery stores are located within a 3km radius from Jean Moulin residence?
- Where is the post office? What are the best restaurants in Le Creusot? Where can I buy flowers?

2 Compilation and testing

2.1 Compilation

The compilation of the software will have a following sequence:
- First unzip the folder and copy the content in the destination folder
- Navigate in the folder directory to find a folder named “Map_loader”. All project files are located inside this folder.
- Click on “map_loader.pro” and give a time to Qt so that Qt can open the project and arrange them in the project navigation
- Your project navigation after this step shall look like figure.1.
- Click on “map.cpp”. This file contains all the functions developed for this project, however note that the file shall be added to each project source always with “map.h” where all functions were declared which will be discuss later.
- Now go to your project folder and copy and paste the directory where you put “map_loader.pro”.
- Paste the complete directory in front of the variable named “static string ProjectDirectory” as highlighted in figure.1.dont forget to change all “/” or “\” to “//”. It is strongly recommended not to use space in directory, which might create problem during the compilation.
- Now, it is the time to add OpenCV compiled library to the project resources. You can download from the GitHub (https://github.com/the address will be send by email). The OpenCV source compiled for Qt5.13.1 installed on Windows 10 OS (Win 10 Pro, revision 2018). If you are using different revision of Qt or your operation system is not complying with above information you need to go to section 4 and read under 4.1 There, it has been explained in detail how to compile OpenCV library for other type or OS or different Qt revision. If you don’t want to follow this project and you just want to know how to use OpenCV in Qt IDE it is worthwhile to have a look at section 4.
- When you have download or compiled the correct OpenCV revision unzip it to somewhere in your computer, the root directory of the library is not important but many developer recommend unzipping in C directory. Now you have to add it to your project resource in the Qt. You have to change your setting in “map_loader.pro” file. This change includes adding additional library and includes file.
- See section 4 for more detail on adding OpenCV to the project resources. The project set file, “map_loader. Pro”, should like figure.2 after completing the set file change.

2.2 Testing and evaluation

- Now you are ready to compile the project and run it. It is straight forward as any Qt project, the only difference is that if you came across with any unknown bug and you run the project in the debug mode pay attention that running shall stop. Otherwise, you will receive OpenCV and OS error. To resolve this issue, you have to close Qt and press Ctrl+ Alt+ Delete combination buttons. The task manager windows will appear. In the windows, click on process tab and look carefully to the active program list. If OpenCV or

map_loader are still running then kill them by end tasking. Re-open the project and built again. It is strongly recommended not to run the project in debug mode.

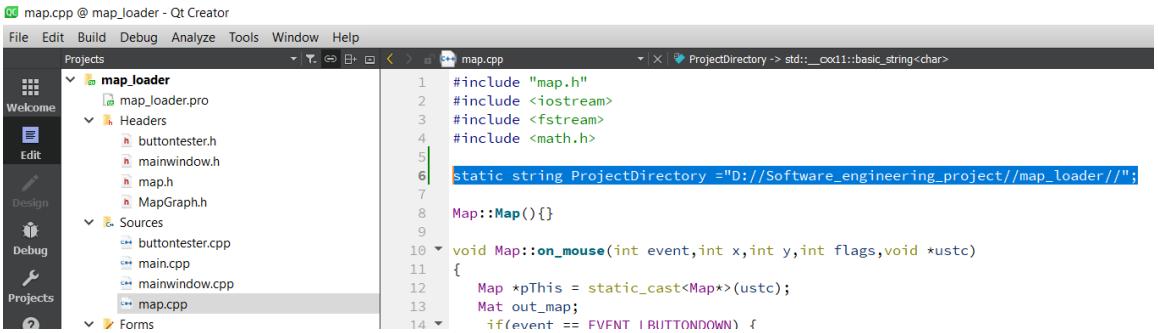


Figure.1. Project directory setting

```
FORMS    += \
    mainwindow.ui

INCLUDEPATH += D:\OpenCV_build\install\include

LIBS += D:\OpenCV_build\bin\libopencv_core412.dll
LIBS += D:\OpenCV_build\bin\libopencv_highgui412.dll
LIBS += D:\OpenCV_build\bin\libopencv_imgcodecs412.dll
LIBS += D:\OpenCV_build\bin\libopencv_imgproc412.dll
LIBS += D:\OpenCV_build\bin\libopencv_features2d412.dll
LIBS += D:\OpenCV_build\bin\libopencv_calib3d412.dll

# more correct variant, how set includepath and libs for mingw
# add system variable: OPENCV_SDK_DIR=D:/opencv/opencv-build/install
# read http://doc.qt.io/qt-5/qmake-variable-reference.html#libs

#INCLUDEPATH += $$(OPENCV_SDK_DIR)/include

#LIBS += -L$$(OPENCV_SDK_DIR)/x86/mingw/lib \
#        -lopencv_core320        \
#        -lopencv_highgui320     \
#        -lopencv_imgcodecs320   \
#        -lopencv_imgproc320     \
#        -lopencv_features2d320  \
#        -lopencv_calib3d320

DISTFILES +=

```

Figure.2. the find set file (section for OpenCV library). The library by default is located in D:\OpenCV_build\ directory. Without setting LIBs file the project cannot be compiled.

3 Functionality requirement and user expectation

This project includes some working, reporting requirements and more important than that respecting to the ethical code. The main ethical code includes referring other authors or citing their works if their works are used in the development or algorithm. All these requirements will be replicated and the author will detail how these requirements have been respected.

3.1 Project requirement and constraints

• Input /output files: the points of interest can be loaded/saved/modified by the user
• The result of a request should be displayed graphically within the software (using any strategy you deem valuable (Qt, OpenGL, OpenCV, etc.) and a text version should be exported in the format of your choice
• The user can select the beginning and the end of his itinerary directly from the screen, or by selecting appropriate items using any kind of menus/lists, depending on your windowing system and implementation
• At least, the following locations have to be available in your software
  ◦ IUT and Condorcet (several buildings, library, lab, class rooms, etc)
  ◦ At least 3 grocery stores, 5 restaurants, 3 fast food places, 3 bakeries
  ◦ Cultural places, such as Parc de la verrerie, the theater, the concert room (ARC), and any other place you deem important
  ◦ at least 20 other shops (hair cut, sportwear, clothes)...
• Apart from the buildings, major roads and streets have to be present. The program should distinguish if you can walk and/or drive. For instance, you cannot drive a car into the Parc de la Verrerie, but you can walk from IUT to Condorcet even if you follow a road...
• You can use google map to create some images offline to spare you some time, but your program must be usable offline: it MUST NOT send requests to Google (or any other server). In other words, you have to create your own map, not interface the existing one.

3.2 Deliverables and Assessment

3.2.1 Deliverable material

You should provide functional software with a graphical user interface implemented in C+ + under QT IDE. The deliverables include, at least, the following items:
• Complete source code + libraries + makefile and any help file to compile your program.
• Report in .pdf format, written using LaTeX
• Defense presentation, 20 min (*.ppt, *.odp, *.pdf format): Last week of December
• All deliverables, except the defense presentation, should be returned by Sunday December the 15th

3.2.2 Assessment and evaluation

You will be assessed on the following items

• Report, no page limit • Intermediate report(s) if any, and project management. (do not forget to link your github repository and trello board if you use any)
• Problem analysis: Critical choices of structures, classes, and their implementation • Strategy for building the user requests and their interpretation
• Path computation algorithms
• User interface. Easiness of use (menus, mouse clicks, various options, etc)
• Visualisation
• Powerpoint Presentation of your work (end of semester) and demo
• Originality and optional (but useful) work: if you think you can go further, do it. These options will be considered as bonuses in the final grade of your project

3.3 Ethics and advices

• If you use or adapt some source code from internet, mention it.
• Groups of 4 students per project at most
• The list (cartography) of the required items can be done in collaboration with all the students
• To obtain the accurate locations of buildings and texture map, use google earth, see illustration next page.
• This project is ambitious and requires some deep reflection before coding
  ◦ Take the time to prepare and sharpen your analysis.
  ◦ Manage your time efficiently
  ◦ Do not focus to much on specific details if something is not fully working
  ◦ Do not wait the very last weeks to start working on your project
  ◦ In case of problem, for any question regarding this project, or if you want to have some feedback on your project while working on it, do not hesitate to contact me at yohan.fougerolle@u-bourgogne.fr

4 Tools and libraries

4.1 Visualization library and display method

Nowadays there are many graphic and visualization libraries have been developed which provide very sophisticated tools, functions and utilities. These libraries constitute the backbone of majority of image processing, mapping and visualization tools and software. For instance, Qt itself has developed a set of tools only for map manipulation and uses different type of displays and visualization kit [xx]. Some libraries such as OpenCV have many different utilities and functions to visualization, which can be used for map manipulation as well. The good thing with OpenCV is that the library has already been used by many scientific communities and companies and it became almost bug free and using CMake as one of the most powerful cross-platform compiling tool facilitate using any libraries in different IDEs. This section is all about how to use these tools to develop the software.

4.2 How to setup Qt and OpenCV on Windows

This instruction is all about how to compile, setup and use OpenCV library in Qt IDE on a Windows platform. The instruction shows how to install Qt, build OpenCV using Cmake and Qt, and run a basic OpenCV example in Qt. This article assumes Windows 10 (Win 10 Pro Revision 2018) has just been installed and fully operational. In terms of hardware resources, as per minimum requirement, this procedure requires close to 10GB of disk space. Approximate disk space required for this procedure is as follows:

```
Qt: 5.06GB
opencv: 522MB
opencv-Build: 3.95GB
downloads:152MB
```

This article uses information from the following pages:
http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html?highlight=installation http://www.laganiere.name/opencvCookbook/chap1s1_2.shtml.
The author strongly recommends reading the article prior proceeding with this procedure.

4.2.1 Qt

1. Installation

This guide is actually originally explained for Qt 5.12.2 with MinGW 7.3.0 and OpenCV 4.0.1. However there is no really changing too much when the revision of Qt and MinGW or OpenCV is changing. If any of the above changed then just you need to compile the library again but the setting and step that a developer shall go through will remain almost unchanged. Except you change the operation system which may need other modifications. In this project, a student distribution of Qt 5.13.1 was used to install and develop, any way it does not change the procedure two much if we stay on windows

platform. If you don't have any Qt source file just download the Qt installer from www.qt.io, and then choose "Download now". This will then download qt-unified-windows-x86-2.0.5-online.exe. Execute the program, and then choose the following settings:

```
Welcome to the Qt online installer: next
Qt Account - your unified login to everything Qt: skip
Setup-Qt: next
installation folder: D:\Qt
select components: Qt-Qt5.9-MingGW 5.3.0 32 bit
select components: Qt-Tools-MinGW 5.3.0
License Agreement: agree and next
start menu shortcuts: next
ready to install: install
```

2. Testing

After installation, Run D:\Qt\Tools\QtCreator\bin\qtcreator.exe

```
File-New file or project-Qt Widgets Application-choose
enter a name and a location: next
select all kits: next
Class information: MainWindow (defaults): Next
Project management: Finish
```

Now a new project is made and ready to start debugging by choosing.

```
Debug-Start Debugging-Start debugging (F5)
```

Now the Qt tab in the Windows taskbar should turn into a progress bar. After some time a new empty window should pop up. Stop debugging either by pressing the Red Cross in the top right of this new window, or choose

```
Debug-Stop debugging
```

The reader is strongly suggested to go to the Qt website and download one of the application and compile and run it to be sure Qt is correctly installed and working properly. Here, https://doc.qt.io/archives/3.3/tutorial1-01.html, you will find an easy tutorial to make the first programing, compilation and test in Qt IDE.

3. Adjust Qt

When you need to add, remove or update a component of Qt, running this program, D:\Qt\MaintenanceTool.exe.

```
maintain Qt: Qt Account: Skip
Setup Qt: Add or remove components: Next
Select components:
next :update
```

4.2.2 CMake

CMake is a cross-platform free and open-source software tool for managing the build process of software using a compiler-independent method. It supports directory

hierarchies and applications that depend on multiple libraries. It is used in conjunction with native build environments such as Make, Qt Creator, Ninja, Apple's X-code, and Microsoft Visual Studio. It has minimal dependencies, requiring only a C++ compiler on its own build system Download CMake from cmake.org. In this guide, 3.7.2 is used, however as explained above using any other revision will not change this procedure significantly. Start cmake-3.7.2-win64-x64.msi, and then choose the following setting:

```
Welcome to the CMake etup Wizzard: next
End-User License Agreement: [X] Accept and next
Install options: [X] Add CMake to the system PATH for all users, next
Destination folder: C:\Program Files\CMake (default), next
Ready to install CMake, Install
```

Cmake is very strong tools and nowadays all developer shall have an idea how does it work. If you don't know this tool it will be worthwhile to look at https://cmake.org/. You will find plenty of resources and tutorial, which explain how to install and use Cmake.

4.2.3 OpenCV

1. Getting OpenCV

Now everything is ready to compile OpenCV for windows. Download OpenCV from any sourceforge or from https://opencv.org/releases/. In this guide, both version 4.1 and 3.2.0 were used and tested. Start opencv-3.2.0-vc14.exe and let it extract to D:\. Now the folder d:\opencv is created and ready to be used.

2. Add MinGW to the windows PATH variable

Before proceeding for compiling, we need to add MinGW to PATH by setting the operating system environment variable:

```
Open the control panel,
System and Security,
System,
Advanced system settings,
Environment Variables,
System Variables,
Variable Name: Path
Variable value: ;D:\Qt\Tools\mingw530_32\bin
```

3. Compiling OpenCV

Start C:\Program Files\CMake\bin\cmake-gui.exe then choose the following settings:

```
Where is the source code: D:\opencv\sources
where to build the binaries: D:\opencv-build
```

Then click Configure, let Cmake create the build directory, and choose the following settings:

```
Specify the generator for this project: MinGW Makefiles
Specify native compilers, next
Compilers C: D:\Qt\Tools\mingw530_32\bin\gcc.exe
Compilers C++: D:\Qt\Tools\mingw530_32\bin\g++.exe
Finish
```

```
Check the box [X]WITH_QT
Check the box [X]WITH_OPENGL
set QT5_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5
```

```
Uncheck the box []ENABLE_PRECOMPILED_HEADERS
```

Then click configure again.

```
Set QT_MAKE_EXECUTABLE to D:\Qt\5.9\mingw53_32\bin\qmake.exe
Set Qt5Concurrent_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5Concurrent
Set Qt5Core_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5Core
Set Qt5Gui_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5Gui
Set Qt5Test_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5Test
Set Qt5Widgets_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5Widgets
Set Qt5OpenGL_DIR to D:\Qt\5.9\mingw53_32\lib\cmake\Qt5OpenGL
Set CMAKE_BUILD_TYPE to Release or RelWithDebInfo
```

Then click configure again Then click generate.
Next open cmd, and type the following commands. To speed up the compile, the -j flag can be used to run multiple compile jobs simultaneously. On an 8 core CPU, you can set it to 8 or higher, so all cores are used. On a core i7 with 8GB ram, the compile takes about 6 minutes.

```
d:
cd d:\
cd opencv-build
mingw32-make -j 8
mingw32-make install
```

If, in the file opencv/sources/modules/videoio/src/cap_dshow.cpp, you have the following error: 'sprintf_instead_use_StringCbPrintfA_or_StringCchPrintfA' was not declared in this scope ...
Try this: put the following line: #define NO_DSHOW_STRSAFE, before the line: #include "DShow.h"
If you have the error: ‘nullptr’ was not declared in this scope.. try this: in Cmake check the box ENABLE_CXX11
If, in the file modules\videoio\src\cap_msmf.cpp you have the error: using invalid field '{anonymous}::ComPtr<T>::p'.. Try this: in Cmake unchecking WITH_MSMF

4.3 Windows 10, Qt 5.9, OpenCV 3.2.0

In this step, we will learn how to use the library in Qt and compile for win 10 platform. So, we assume either the OpenCV library is available or built in section 5.3.

1. Add OpenCV compiled libraries to the windows PATH variable

Now, everything is ready to use OpenCV, just before starting we have to add library location to the path variable so as complier can use the library during compilation. Set following parameters:

```
Open the control panel,
System and Security,
System,
Advanced system settings,
Environment Variables,
System Variables,
Variable Name: Path
Variable value: ;D:\opencv-build\install\x86\mingw\bin
```

4.3.2 Compile and run the example

To use OpenCV in a read program now run Qt by going to the following directory and Running D:\Qt\Tools\QtCreator\bin\qtcreator.exe. Create your first program by setting following parameters.

```
File-New file or project-Qt Widgets Application-choose
enter a name and a location: next
select all kits: next
Class information: MainWindow (defaults): Next
Project management: Finish
```

Now a new project is made. Modify the .pro file like this:

```
#-------------------------------------------------
#
# Project created by QtCreator 2017-03-05T12:30:06
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = opencvtest
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which as been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0


SOURCES += main.cpp\
        mainwindow.cpp

HEADERS  += mainwindow.h

FORMS    += mainwindow.ui

INCLUDEPATH += D:\opencv\build\include

LIBS += D:\opencv-build\bin\libopencv_core320.dll
LIBS += D:\opencv-build\bin\libopencv_highgui320.dll
LIBS += D:\opencv-build\bin\libopencv_imgcodecs320.dll
LIBS += D:\opencv-build\bin\libopencv_imgproc320.dll
LIBS += D:\opencv-build\bin\libopencv_features2d320.dll
LIBS += D:\opencv-build\bin\libopencv_calib3d320.dll

# more correct variant, how set includepath and libs for mingw
# add system variable: OPENCV_SDK_DIR=D:/opencv/opencv-build/install
# read http://doc.qt.io/qt-5/qmake-variable-reference.html#libs

#INCLUDEPATH += $$(OPENCV_SDK_DIR)/include

#LIBS += -L$$(OPENCV_SDK_DIR)/x86/mingw/lib \
#        -lopencv_core320        \
#        -lopencv_highgui320     \
#        -lopencv_imgcodecs320   \
#        -lopencv_imgproc320     \
#        -lopencv_features2d320  \
#        -lopencv_calib3d320
```

And modify mainwindow.cpp like this:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // read an image
    cv::Mat image = cv::imread("f://1.jpg", 1);
    // create image window named "My Image"
    cv::namedWindow("My Image");
    // show the image on window
    cv::imshow("My Image", image);
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

This is all needed to compile and use the OpenCV library in the program.

5 Map modality and information visualization

5.1 Loading different map modality

The first step of development is to have a function to load a map. For that, every map is considered as an image. Since, we are using OpenCV, the tool and functions are already in the tool and we just need to recognize and use them. Different map of Le Creusot city were cut from Google map and saved in the project repository. Then using OpenCV IMREAD function the map is loaded and displayed. The function has different setting. I have set the parameter to have a navigation tools and a little menu on the display dialog box. This way, the user can save the results of the search. Figure.3 shows one map loaded this way. This initial map can have certain level of resolution so zoom in and zoom out of the map will depend on the initial quality.

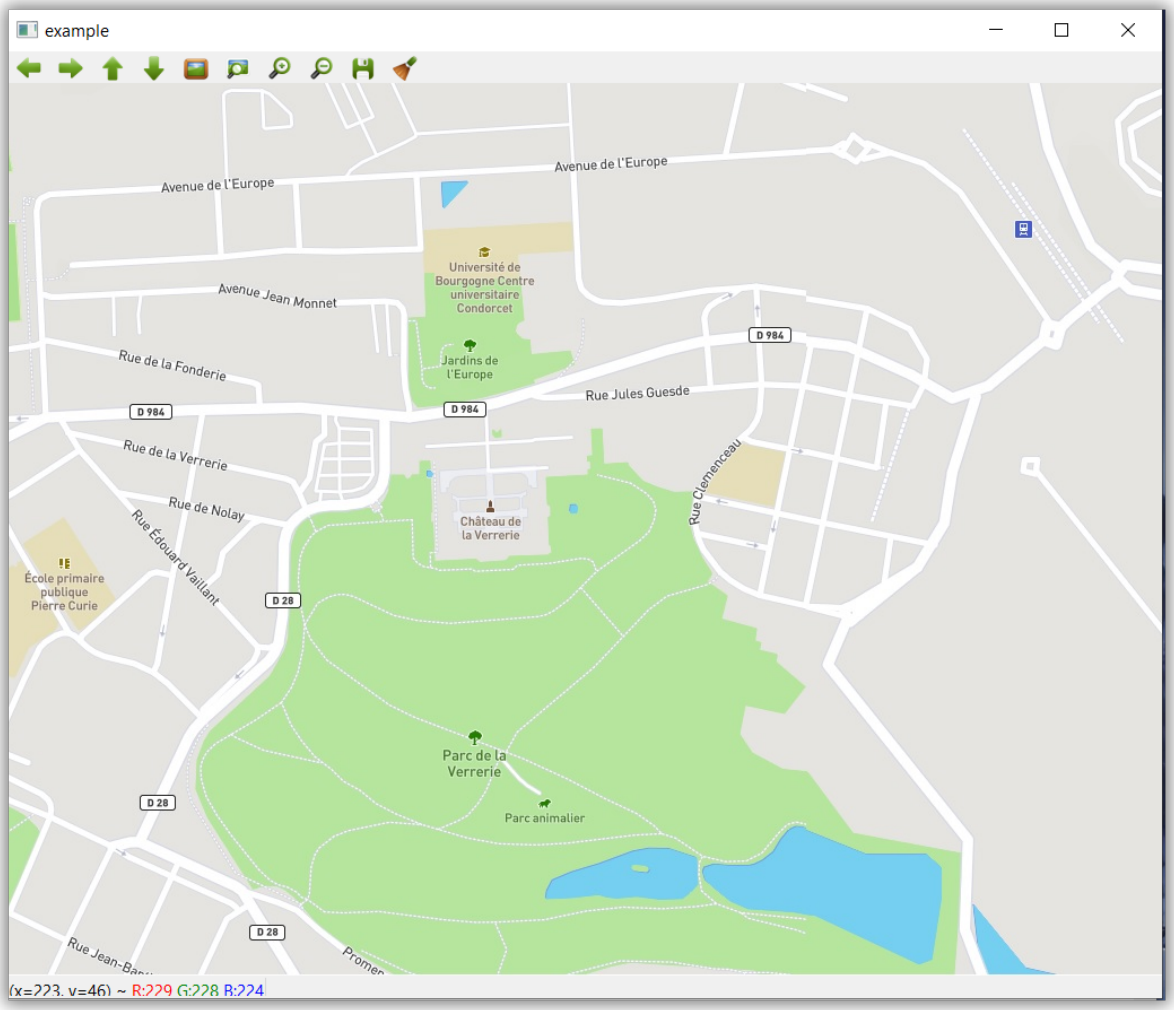


Figure 1: transpiration image modality

Map image with different zoom level. Other Map types also can be loaded for instance topological or satellite map. But they have to be produced by online map software such as Google Map and stored in the project repository. Figure 4 and 5 shows other map modalities.

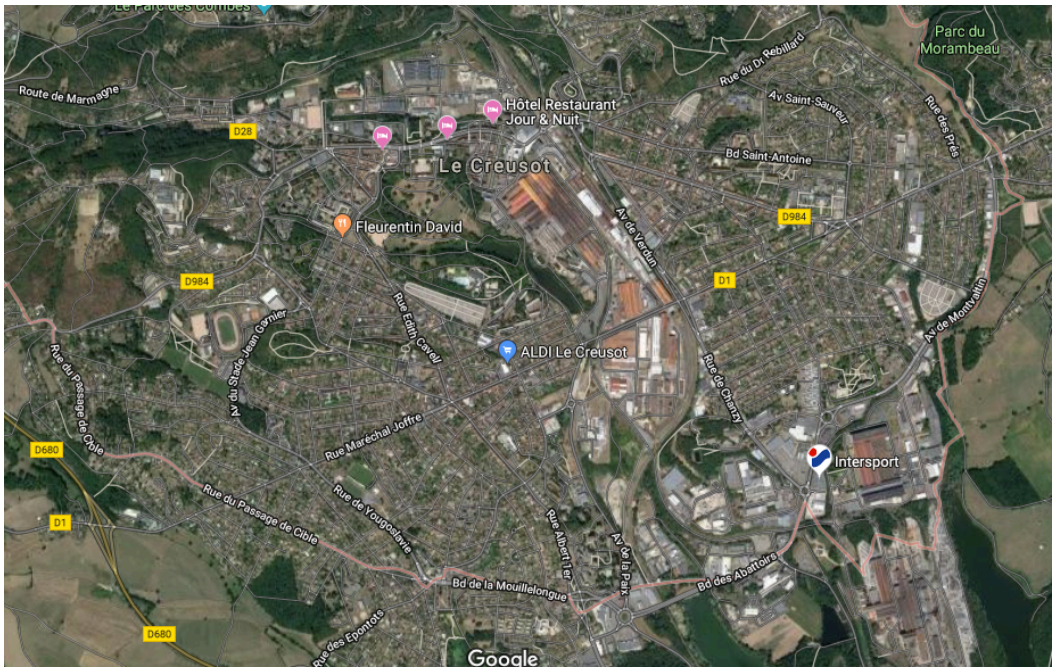


Figure 4: satellite Map

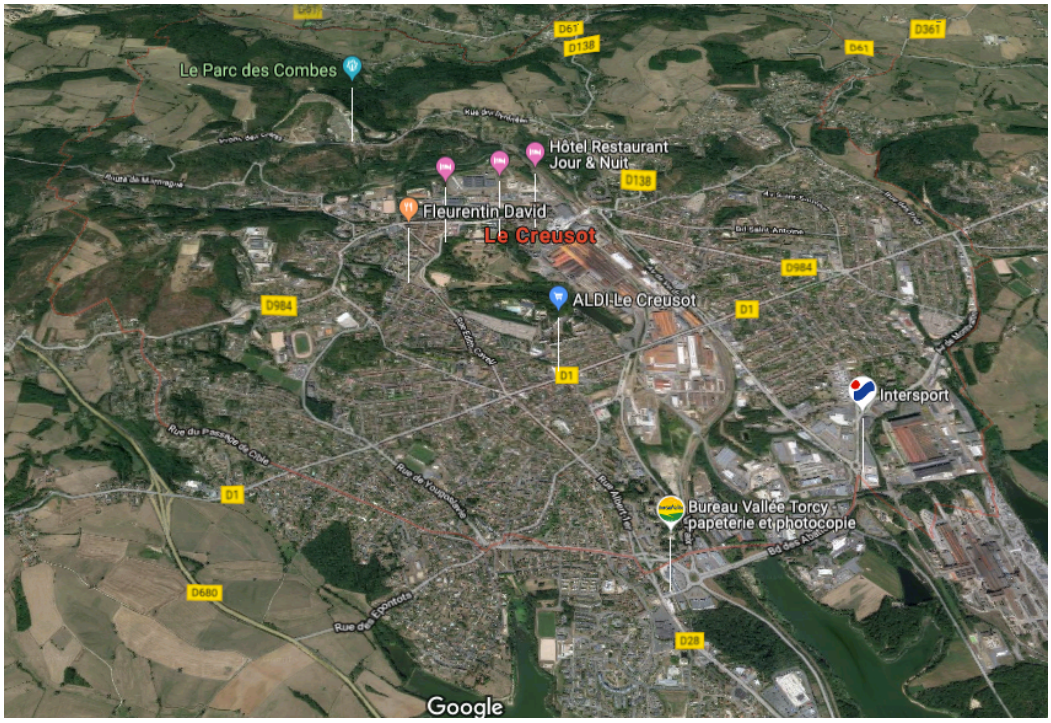


Figure 5: 3D map

The number of the map which can be open at once in this tool is not limited to one and depending on hardware capacity up to limited number of map can be loaded at once by different function thanks to OpenCV, as shown in Figure 6.

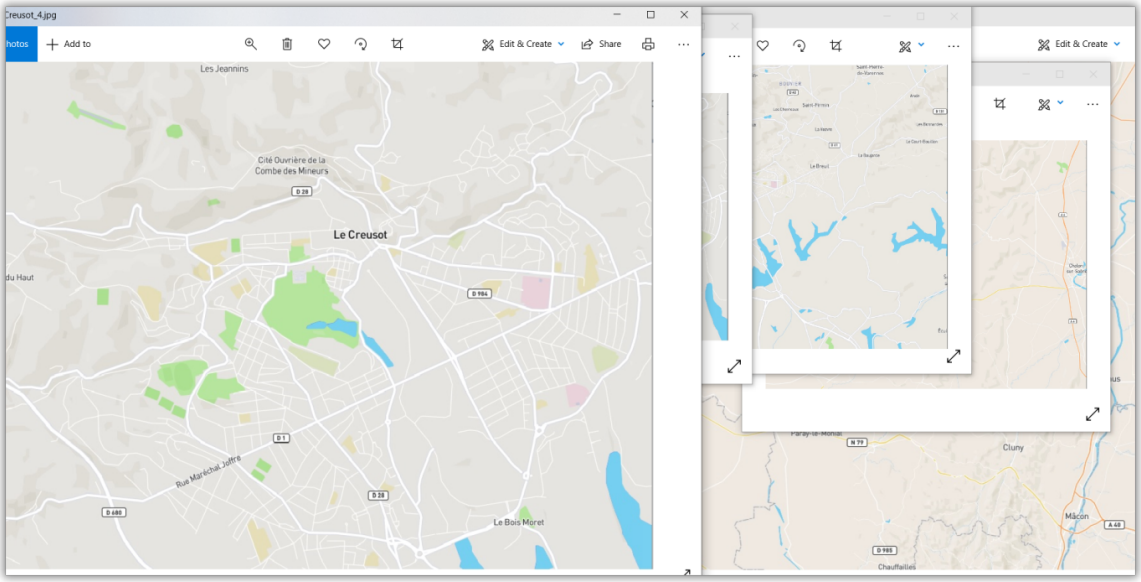


Figure 6. Several maps has been displayed and manipulated at once

5.2 Location marker

In this development, we can use different marker to show a given address on the map, for instance a historical site or a shopping market. But during the development and test phase only few of them as shown in Figure.7 were used. However, the number of the marker and the type never been limited by our design and user can select his own marker and loaded. The end-user needs to select his marker and make an image and store in the repository and call it. Here in this development, we use simple circle marker for path planning and itinerary indication and other type of marker for individual location.

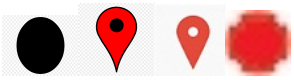


Figure 7. Different types of maker used to visualize paths and different points.

One dedicated function was developed to overlay the marker on the default map and prepare the final map for display. This function will be introduced later when the implementation will be explained. There is a function to put the text on the map. For instance, if require to display the name of the location on the map it is possible.

5.3 Path visualization and selected location display

Our design shall have sort of facilities to allow the user or developer to select the point and get the coordinate for a given map. These facilities are using two windows, in one

window the point is selected by clicking on the map and in the other windows the point is displayed. As shown in Figure.8, a marker is displayed on the point selected by the user. This function is important because it gives immediate feedback to the user if the coordinate returned is correct or not. If it is not correct and the selected point is displayed in different location, the user or the developer can use this feedback to debug the program.

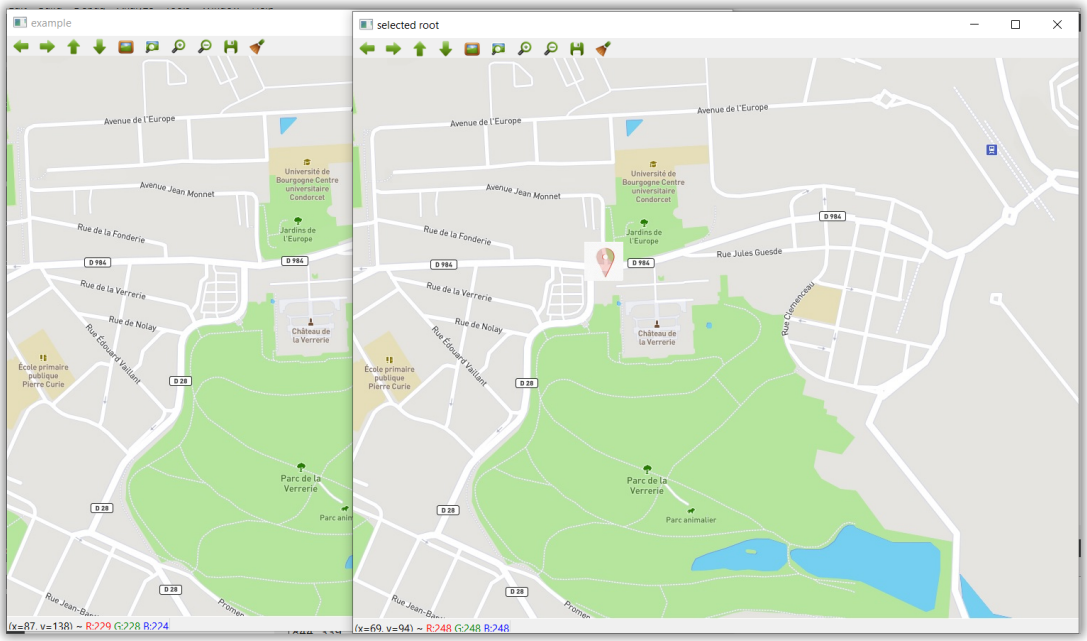


Figure 8. A marker is inserted at the selected location as immediate visual feedback

The following windows show individual point calculated by path planning function and a complete path displayed circle maker. As seen the marker is different in Figure 8 and 9. The left image shows the makers and right image is a map where is mouse click is used to select the coordinate of the location.

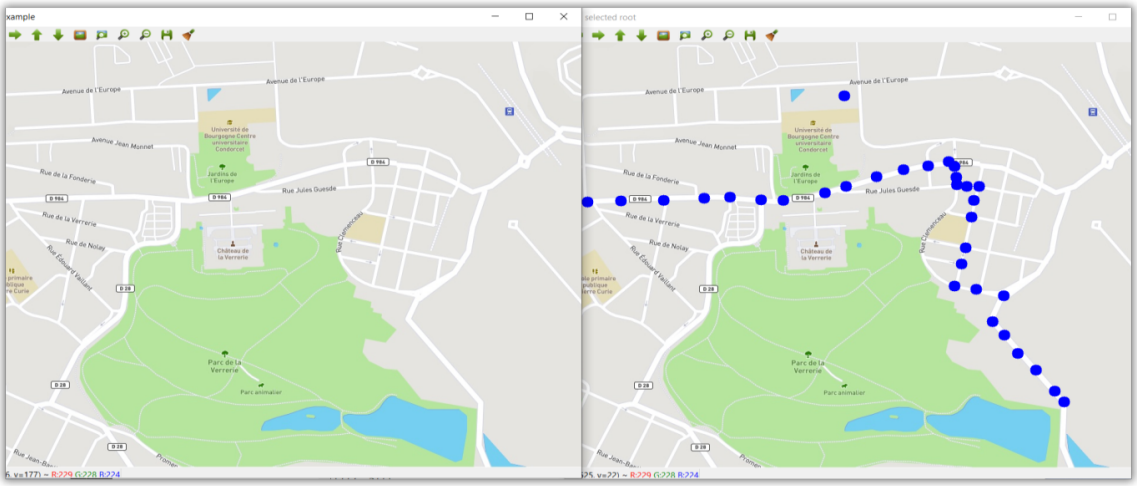


Figure 9. Calculated path between two or more selected point

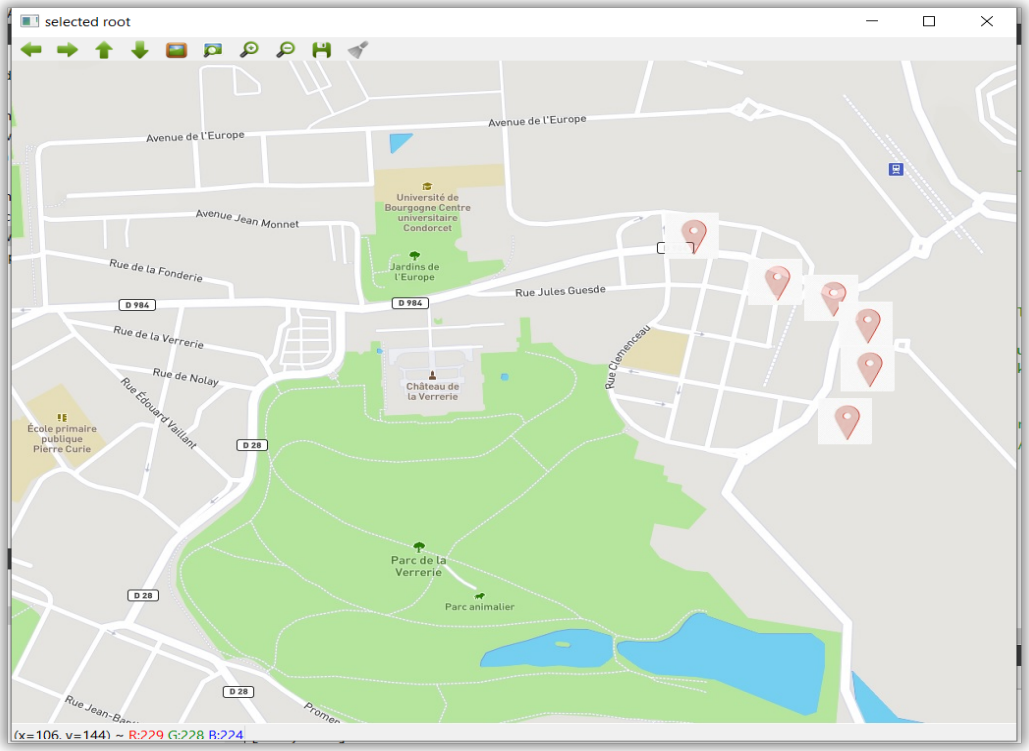


Figure 10. Example of path planning between two given point with different marker

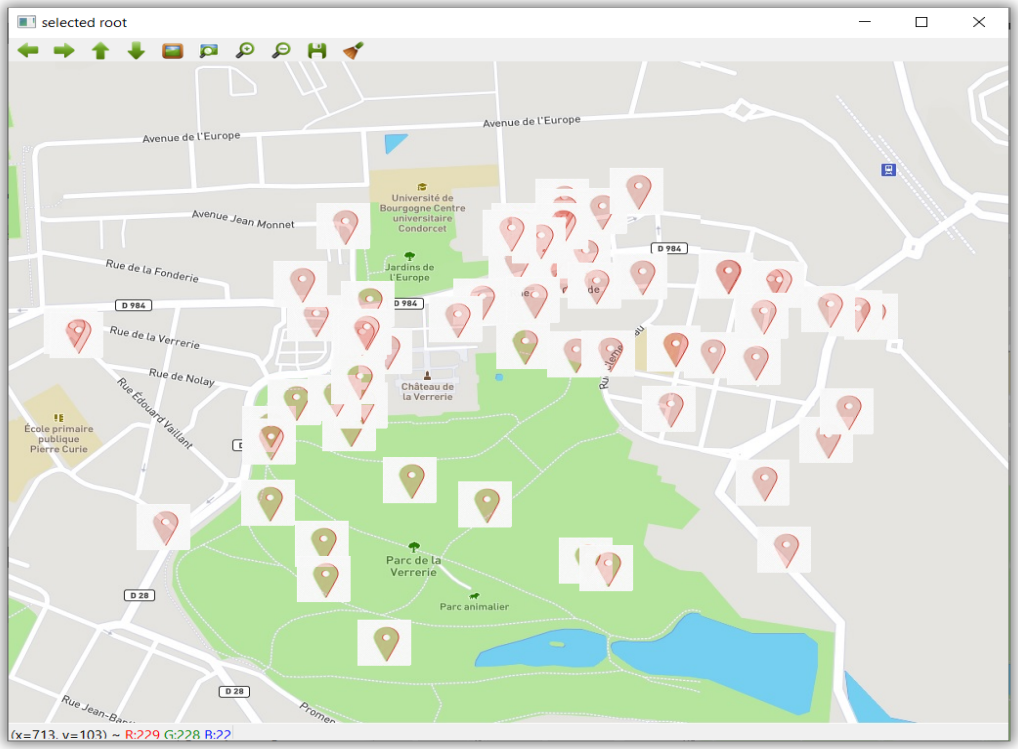


Figure 11. All selected points to make adjacent matrix and neighborhood matrix is shown

All selected points are visualized on the city map just for visualization purpose. This is map of randomly selected points. Point selected on the paper map and the adjacent matrix was extracted for the point shown in Fig.12. The path planning and trajectory selection between two given points and via third point algorithm will be discussed in the next section.

Figure 12. Actual point selected on the map to make neighborhood matrix and path

6 Algorithm

This section is dedicated to shortest path planning, which includes, given a graph and a source vertex in the graph, find shortest paths from source to all vertices in the given graph. This implicitly means how we can find a shortest path between two points and repeating this algorithm two times from source vertex and intermediate point and intermediate point and destination point will lead to going from point A to C through B. the main shortest path algorithm is called “Prim’s algorithm for minimum spanning tree”. In the Prim’s MST, a SPT (shortest path tree) is defined with given source as root. The algorithm maintains two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree. At every step of the algorithm, a vertex, which is in the other set (set of not yet included), is found which has a minimum distance from the source. Below description are the detailed steps used in the algorithm to find the shortest path from a single source vertex to all other vertices in the given graph. If the shortest path between two point is expected then only that path is selected.

6.1 Algorithm steps

1) Create a set of SPT set that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
3) While SPT doesn’t include all vertices
    a) Pick a vertex u, which is not there in SPT and has minimum distance value.
    b) Include u to SPT
    c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

6.2 Algorithm with an example

Let us assume following graph to understand above step with an example:

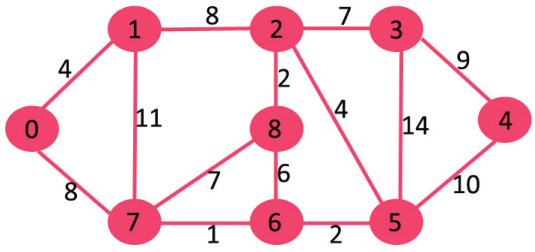


Figure 13. Given example graph

Let assume also u=0 as start vertex. The set *SPT* is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now if we want to pick up a vertex with minimum distance value the vertex 0 will be picked, include it in SPT. So *SPT* becomes {0}. After including 0 to *SPT* we will update distance values of its adjacent vertices. Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8. Following sub-graph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in SPT are shown in Figure 14 in green color.

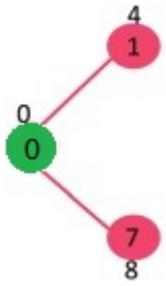


Figure 14. Step 1: Selected path from source vertex to the neighboring vertex

We will pick the vertex with minimum distance value and not already included in SPT. The vertex 1 is picked and added to SPT. So SPT now becomes {0, 1}. Update the distance values of adjacent vertices of 1. The distance value of vertex 2 becomes 12. This detail is shown in Figure 15.

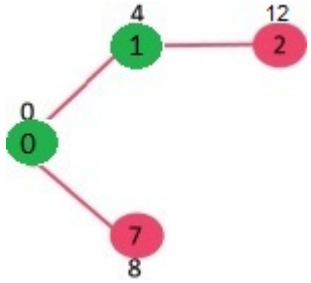


Figure 15. Step 2 of the algorithm

Lets pick the vertex with minimum distance value and not already included in SPT. Vertex 7 is picked. So SPT now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (15 and 9 respectively).

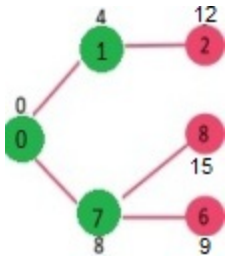


Figure 16: Step 3 of the algorithm

Pick the vertex with minimum distance value and not already included in SPT. Vertex 6 is picked. So SPT now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.

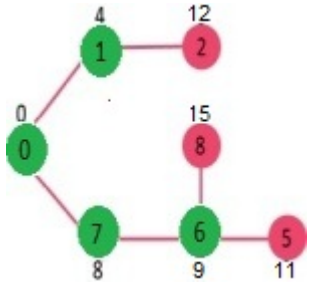


Figure 17: Step 4 of the algorithm

We repeat the above steps until SPT does include all vertices of given graph. Finally, we get the following Shortest Path Tree (SPT).

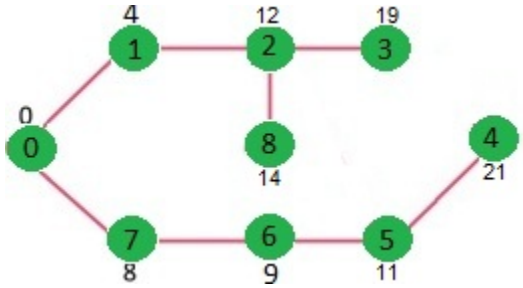


Figure 18: the shortest path is found

6.3 Example code for algorithm

An example algorithm of shortest path calculation is given in this section, however a different revision of the code is used for implementation.

```
// A C++ program for Dijkstra's single source shortest path
// The program is for adjacency matrix representation of the

#include <limits.h>
#include <stdio.h>

// Number of vertices in the graph
#define V 9

// A utility function to find the vertex with minimum distan
// the set of vertices not yet included in shortest path tre
int minDistance(int dist[], bool sptSet[])
{
    // Initialize min value
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

// A utility function to print the constructed distance arra
int printSolution(int dist[])
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}

// Function that implements Dijkstra's single source shortes
// for a graph represented using adjacency matrix representa
void dijkstra(int graph[V][V], int src)
{
    int dist[V]; // The output array.  dist[i] will hold the
    // distance from src to i

    bool sptSet[V]; // sptSet[i] will be true if vertex i is
    // path tree or shortest distance from src to i is final

    // Initialize all distances as INFINITE and stpSet[] as
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

    // Distance of source vertex from itself is always 0
    dist[src] = 0;

    // Find shortest path for all vertices
    for (int count = 0; count < V - 1; count++) {
        // Pick the minimum distance vertex from the set of
        // yet processed. u is always equal to src in the fi
```

```
        int u = minDistance(dist, sptSet);

        // Mark the picked vertex as processed
        sptSet[u] = true;

        // Update dist value of the adjacent vertices of the
        for (int v = 0; v < V; v++)

            // Update dist[v] only if is not in sptSet, ther
            // u to v, and total weight of path from src to
            // smaller than current value of dist[v]
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    // print the constructed distance array
    printSolution(dist);
}

// driver program to test above function
int main()
{
    /* Let us create the example graph discussed above */
    int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                        { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                        { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                        { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                        { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                        { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                        { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                        { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                        { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

    dijkstra(graph, 0);

    return 0;
}
```

7 Development and implementation

The following functions and software were developed based on Qt 5.13.1 with MinGW 7.3.0 and OpenCV 4.0.1. The environmental setting, using Cmake to compile library for Qt has all already explained in section 4 [1]. Some additional guide on using windows of open CV was provided in link [2]. However, the same technique as Qt5.13.1 was applied to compile the library for the last revision of the Qt5. The compiled library was downloaded on the GitHub [3]. Then, based on this library all visualization and coding was setup. The main code of application is seen in the following piece of code.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "map.cpp"

//open CV menus creatation
#include <opencv2/highgui.hpp>

//
//mouse event and data
#include "buttontester.h"
#include <QPushButton>
#include <QVBoxLayout>
//end of mouse
void savedata(int state, void* userdata);
void savedata(int state, void* userdata) {...}                    unused parameter 'state'   unused parameter 'userdata'
//
//void on_mouse(int event,int x,int y,int flags,void *ustc);
/*void on_mouse(int event,int x,int y,int flags,void *ustc) ...*/

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    /* read mouse data*/
   /* ButtonTester *testArea = new ButtonTester; ...*/
    // read an image
   /*     cv::Mat image = cv::imread("C://Users//mirza//Documents//map_loader//1.jpg"); ...*/
   // cv::namedWindow("My Image",WINDOW_AUTOSIZE);
   // createButton("take coordinate and save them",shortestpath);
   // createButton("shortest path",savedata);
    // show the image on window
   // cv::imshow("My Image", image);

   /* cv::Mat org; ...*/
   /* ofstream myfile ("C://Users//mirza//Documents//map_loader//example.txt"); ...*/

    string Map_image ="C://Users//mirza//Documents//map_loader//Le_Creusot_5.jpg";
    string Marker_image ="C://Users//mirza//Documents//map_loader//marker2.jpg";
    Map Le_creusot;
    cv::Mat img;
    calculate_shortest_path("C://Users//mirza//Documents//map_loader//root.txt", &Le_creusot);
    //cv::Mat img= cv::imread("C://Users//mirza//Documents//map_loader//Le_Creusot_5.jpg",1);

    Le_creusot.org=  cv::imread(Map_image,1);
    Le_creusot.marker = cv::imread(Marker_image,1);
    //display_shortest_path(&Le_creusot);
    namedWindow("example",WINDOW_AUTOSIZE);
    createButton("take coordinate and save them", shortestpath);
    createButton("shortest path",savedata);
    cv::setMouseCallback("example", &Map::on_mouse, &Le_creusot );
    imshow("example", Le_creusot.org);
     waitKey();

}

MainWindow::~MainWindow()
{
    delete ui;
}
```

The code does following functionalities:

- Display different format of the local map (in our case Le Creusot Map)
- Put marker (of different type) on the select location
- Write the location coordinate in a text file for later reference
- If new location is selected then it will added to the previous location record and saved in a text file
- All location can be visualized at one to show to what extent the selected locations covers the city map
- For a give root, the path from departure to destination can be visualize
- Two buttons are allocated to shortest path calculation and visualization
- Map can be zoomed in and out
- If the map is too big it can pan by mouse

- And some other functionality which will be detail below

The main class for map and all activities related to map is located in map.h file for declaration and map.cpp for development. First map.h is discussed. The map, marker, member functions, path and other variables are declared in this file. Make a special attention to on_mouse function will uses mouse call back to select the point or register them in the text file.

```cpp
#ifndef MAP_H
#define MAP_H
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
using namespace cv;

#include <opencv2/highgui.hpp>
#include <vector>
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <iostream>
using namespace std;

class Map
{
private:
    //variable
    string m_winname;
    //fuctions

public:
    //variable
    static Point pt;//mouse position;
    cv::Mat org;
    cv::Mat marker;
    vector<Point> path;
    string select_marker_type;
    char coordinate[16];
    //fucntions
    Map();
    static void on_mouse(int,int,int,int,void *);
};
void overlayImage(const cv::Mat&, const cv::Mat&, cv::Mat&, cv::Point2i);
void calculate_shortest_path(string, Map*);
void display_shortest_path(Map*);
void Marker_selector(Map*, string);
void shortestpath(int, void*);
#endif // MAP_H
```

```cpp
#include "map.h"
#include <iostream>
#include <fstream>


Map::Map(){}

void Map::on_mouse(int event,int x,int y,int flags,void *ustc)
{
   Map *pThis = static_cast<Map*>(ustc);
   Mat out_map;
    if(event == EVENT_LBUTTONDOWN) {
        ofstream file("C://Users//mirza//Documents//map_loader//example.txt",std::ios_base::app);//save coordinates to txt file;
        if(!file)
        {
            cout << "open file error!";
            cout<<flags<<endl;
            cout<<&pThis<<endl;
        }
        else
        {
            //cout<<"beging writing"<<endl;
            //pThis->pt = Point(x,y);
            file<<x<<','<<y<<endl;
            cout << "(" << x << ", " << y << ")" << endl;
            file.close();
            //****define the marker type here****
            Marker_selector(pThis, "shape");
            //**********end of marker selection****
            namedWindow("selected root");
            if(pThis->select_marker_type == "circle_marker")
            {
               circle(pThis->org,Point(x,y),10,Scalar(255,0,0),FILLED,LINE_8,0);
               imshow("selected root", pThis->org);
               //pThis->marker.copyTo(pThis->org(cv::Rect(x,y,pThis->marker.cols, pThis->marker.rows)));
            }
            else if(pThis->select_marker_type =="from_file_maker")
            {
                overlayImage(pThis->org,pThis->marker,out_map,Point(x,y));
                imshow("selected root", out_map);
            }

            waitKey();
        }
    }
}
```

The definition of on_mouse function is as follows. The function takes the city map class and stores all selected point after each click on the map. So far, there is a possibility to use two types or markers. The markers either are circular type or can be any shape, which will be given as a jpg image. A separate map will be displayed to show when the user select the point. The coordinate of the point will be directly print out in the debug windows. Some information was collected from [4]-[9].
To display marker on the map we need to aggregate the marker into the image matrix. This way of the insertion of a maker inside a given map is done by "overlayImage" function [10][11].

```
void overlayImage(const cv::Mat &background, const cv::Mat &foreground, cv::Mat &output, cv::Point2i location)
{
  background.copyTo(output);


  // start at the row indicated by location, or at row 0 if location.y is negative.
  for(int y = std::max(location.y , 0); y < background.rows; ++y)
  {
    int fY = y - location.y; // because of the translation
    // we are done of we have processed all rows of the foreground image.
    if(fY >= foreground.rows)
      break;
    // start at the column indicated by location,
    // or at column 0 if location.x is negative.
    for(int x = std::max(location.x, 0); x < background.cols; ++x)
    {
      int fX = x - location.x; // because of the translation.
      // we are done with this row if the column is outside of the foreground image.
      if(fX >= foreground.cols)
        break;
      // determine the opacity of the foregrond pixel, using its fourth (alpha) channel.
      double opacity =
        ((double)foreground.data[fY * foreground.step + fX * foreground.channels() + 3])    use of old-style cast   implicit conversion changes signedness: '
        / 255.;
      // and now combine the background and foreground pixel, using the opacity,
      // but only if opacity > 0.
      for(int c = 0; opacity > 0 && c < output.channels(); ++c)
      {
        unsigned char foregroundPx =
          foreground.data[fY * foreground.step + fX * foreground.channels() + c];    implicit conversion changes signedness: 'int' to 'unsigned int'   implic
        unsigned char backgroundPx =
          background.data[y * background.step + x * background.channels() + c];    implicit conversion changes signedness: 'int' to 'unsigned int'   implicit
        output.data[y*output.step + output.channels()*x + c] =    implicit conversion changes signedness: 'int' to 'unsigned int'   implicit conversion chang
          backgroundPx * (1.-opacity) + foregroundPx * opacity;    implicit conversion turns floating-point number into integer: 'double' to 'uchar' (aka 'uns.
      }
    }
  }
}
```

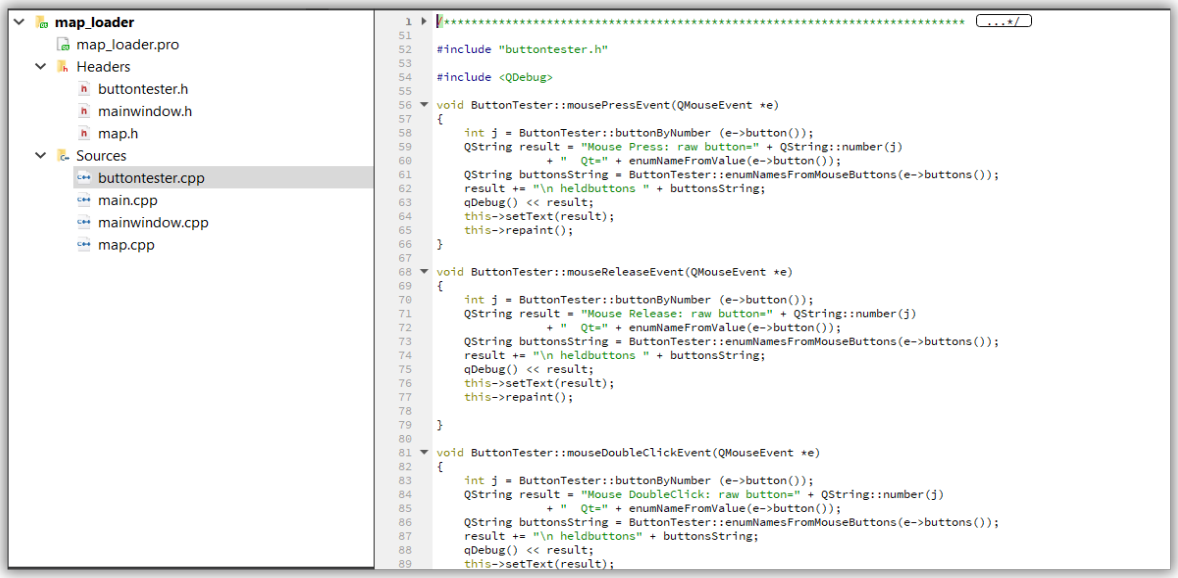
Shortest path between two given points are calculated based on the algorithm given in [12]. The path is selected randomly just for display purpose. But in reality the path shall be calculated given the overall point selected from the map and start and end point, which will be done separately using algorithm [14].

```
void calculate_shortest_path(string filename, Map *city)
{
    string line;
    int x, y;
    char ch;
    unsigned index=1;
    cout<<filename<<endl;
    ifstream path_file (filename);
    if (path_file.is_open())
    {
      while (!path_file.eof())
      {
        path_file >>x>>ch>>y;
        //cout<<index<<endl;
        //cout<<"coordinate read from the file"<<endl;
        cout<<x<<","<<y<<endl;
        city->path.push_back(Point(x,y));
        index=index+1;
      }
      path_file.close();
    }

    else cout << "Unable to open file";
}
```

Sometimes it is needed to visualize the path to test the program and algorithm to be sure it is working properly. The following function is developed for this purpose, based on the selected marker.

```
void display_shortest_path(Map *city)
{
    unsigned size_of_path;
    cv::Mat path_map, path_update;
    unsigned i;
    size_of_path= city->path.size();
    path_update =city->org;
    cout<<"size of path"<<endl;
    cout<<"size of path"<<size_of_path<<endl;
    for(i=2;i<size_of_path;i++)
    {
        cout<<city->path[i]<<endl;
        overlayImage(path_update,city->marker,path_map,city->path[i]);
        path_update = path_map;
    }
    namedWindow("selected root");
    imshow("selected root", path_map);
    waitKey();
}
void Marker_selector(Map *city, string choice)
{
    if(choice == "shape")
    {
        city->select_marker_type = "circle_marker";
    } else
    {
       city->select_marker_type = "from_file_maker";
    }
}
void shortestpath(int state, void* userdata)
{
    Map *pThis = static_cast<Map*>(userdata);
    //cv::ogl::Texture2D backgroundTex = userdata;
    cout<<"print out button value"<<state<<endl;
    cout<<pThis->path<<endl;
}
```

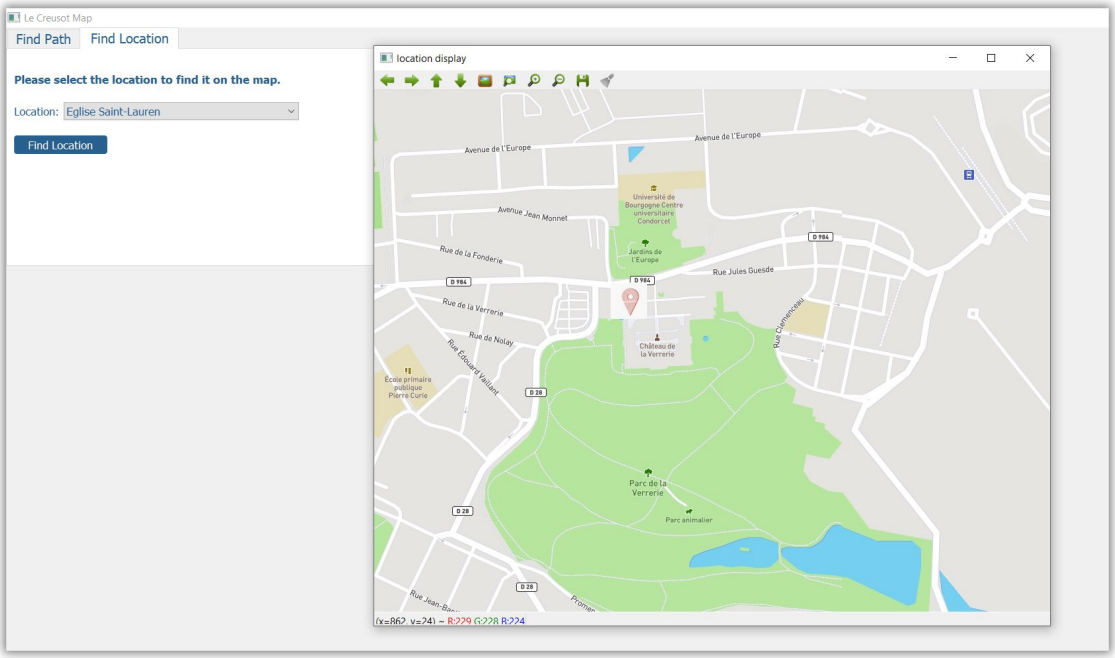
This functions and its related library is used to facilitate mouse events. The detail can be seen in the repository. Some of these functions were used in on_mouse function, which was used in the first part at the top.

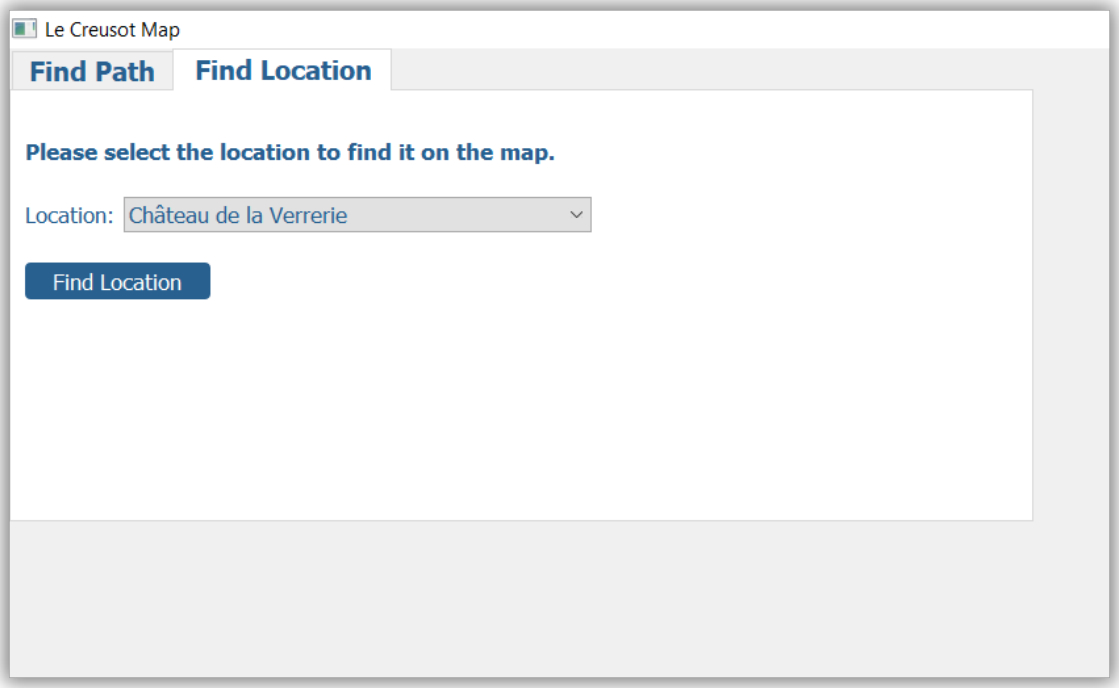


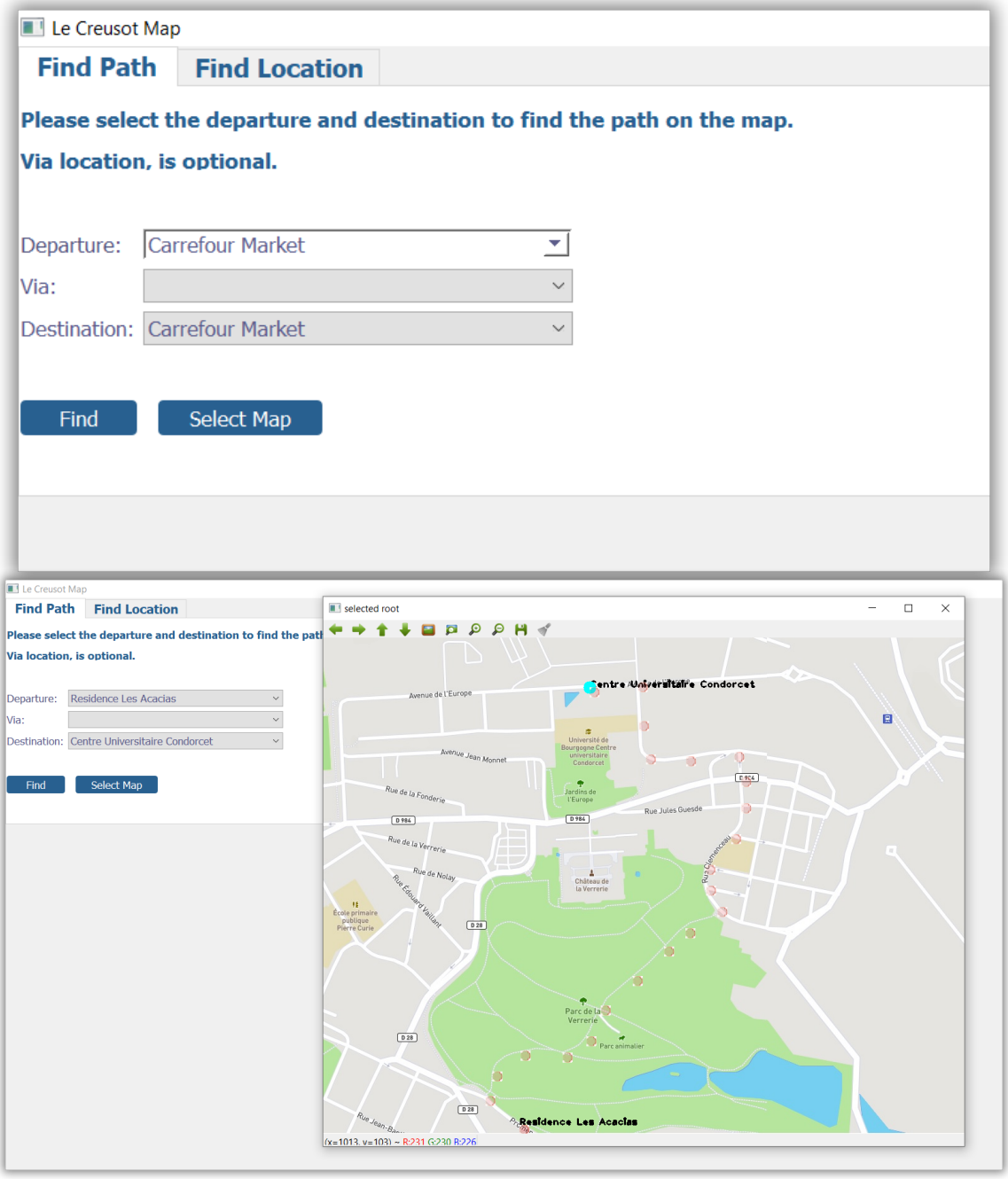

```
/********************************************************************************* ...*/

#include "buttontester.h"

#include <QDebug>

void ButtonTester::mousePressEvent(QMouseEvent *e)
{
    int j = ButtonTester::buttonByNumber (e->button());
    QString result = "Mouse Press: raw button=" + QString::number(j)
                + "  Qt=" + enumNameFromValue(e->button());
    QString buttonsString = ButtonTester::enumNamesFromMouseButtons(e->buttons());
    result += "\n heldbuttons " + buttonsString;
    qDebug() << result;
    this->setText(result);
    this->repaint();
}

void ButtonTester::mouseReleaseEvent(QMouseEvent *e)
{
    int j = ButtonTester::buttonByNumber (e->button());
    QString result = "Mouse Release: raw button=" + QString::number(j)
                + "  Qt=" + enumNameFromValue(e->button());
    QString buttonsString = ButtonTester::enumNamesFromMouseButtons(e->buttons());
    result += "\n heldbuttons " + buttonsString;
    qDebug() << result;
    this->setText(result);
    this->repaint();

}

void ButtonTester::mouseDoubleClickEvent(QMouseEvent *e)
{
    int j = ButtonTester::buttonByNumber (e->button());
    QString result = "Mouse DoubleClick: raw button=" + QString::number(j)
                + "  Qt=" + enumNameFromValue(e->button());
    QString buttonsString = ButtonTester::enumNamesFromMouseButtons(e->buttons());
    result += "\n heldbuttons" + buttonsString;
    qDebug() << result;
    this->setText(result);
```

8 Testing and some results

The development was tested in the Qt IDE and the result will be presented by Demo. The GUI and some results were presented in Figure 19.






Le Creusot Map
Find Path
Find Location
Please select the departure and destination to find the path on the map.
Via location, is optional.
Departure:
Carrefour Market
Via:
Destination:
Carrefour Market
Find
Select Map
Le Creusot Map
Find Path
Find Location
Please select the departure and destination to find the pat
Via location, is optional.
Departure:
Residence Les Acacias
Via:
Destination:
Centre Universitaire Condorcet
Find
Select Map
selected root
Centre Universitaire Condorcet
Avenue de l'Europe
Avenue Jean Monnet
Rue de la Fonderie
Rue de la Verrerie
Rue de Nolay
Rue Jules Guesde
Château de la Verrerie
Parc de la Verrerie
Parc animalier
Residence Les Acacias

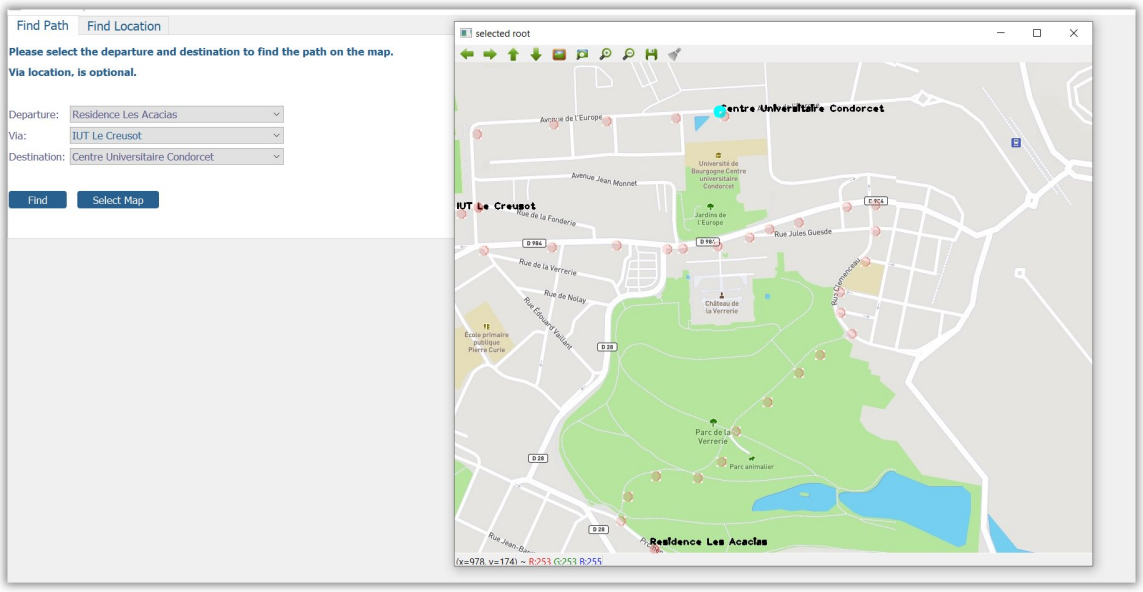


Figure 19: the GUI and some result of the location indication and shortest path

9 Future work and improvement

A lot of effort was made to generalize the development so that to develop a set of tool totally for MAP manipulation and visualization. This part took a lot time to implement as more general solution, although it can be significantly improved. The current online approaches use remote server and strong database to calculate the map, which might not help travels when they are lacking any kind of Internet connection.

In the project, the adjacent matrix of the selected points were made by hand due to lack of time. It seems it will be very challenging to do this automatically especially when we are trying to add new point to our cloud and update the graph because this function can not be fully automatic and some how human intervention is expected, at least in the verification level (to be sure the final result will be enough interesting to be used by any end user during daily life). Using smart phones are very dominant nowadays and it seems every application will spread all around if target for smart phone. So this specification for desktop application might be very interesting also for smart phone application.

So the next challenges can be defined as:

- Developing a GUI and MAP manipulation software kit for smart phone
- Develop an interface to build adjacent matrix in a semi-automatic way decreasing the load of human intervention and producing high precision graph
- Using smart phone GPS data as an input for graph update function
- The ultimate object can be deriving graph from MAP directly by MAP image processing technic (line detection, corner detection and so on)

10 References:

[1] https://wiki.qt.io/How_to_setup_Qt_and_openCV_on_Windows
[2] https://docs.opencv.org/2.4/modules/highgui/doc/qt_new_functions.html
[3] https://bitbucket.org/DJNing/cpp_project/src/master/
[4] http://www.cplusplus.com/doc/tutorial/files/
[5] http://www.cplusplus.com/forum/beginner/8388/
[6] http://www.cplusplus.com/reference/vector/vector/at/
[7] https://stackoverflow.com/questions/7868936/read-file-line-by-line-using-ifstream-in-c
[8] https://stackoverflow.com/questions/6406356/how-to-write-vector-values-to-a-file
[9] https://docs.opencv.org/master/db/deb/tutorial_display_image.html
[10] http://jepsonsblog.blogspot.com/2012/10/overlay-transparent-image-in-opencv.html
[11] https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html
[12] https://docs.opencv.org/3.4/d3/d96/tutorial_basic_geometric_drawing.html
[13] https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html
[14] https://www.geeksforgeeks.org/c-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/