Installation manual for the segmentation of the aorta artery project (cpPlugins + CreaTools)

Sergio Daniel Hernández Charpak

Advisors:
Marcela Hernndez Hoyos[1]
Leonardo Flórez[2]
Eduardo Dávila[3]

January 22, 2017

Universidad de los Andes
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas y Computación
Bogotá, Colombia

[1]Universidad de los Andes, Bogotá, Colombia
[2]Pontificia Universidad Javeriana, Bogotá, Colombia
[3]Laboratoire CREATIS, Lyon, France

1 Installation manual

The following installation instructions are for a Linux platform operating system. They are tested for Ubuntu 14.04 from Canonical. Working versions for other operating system platforms such as Microsoft Windows and Apple MacOS are currently under development as there are still some bugs to fix. The project uses CMake as compilation helper. It should eventually compile on the three major box flavors (linux, windows, mac).

1.1 Libraries: CMake, VTK, ITK, QT

- CMake (version $\geq$ 2.8)

CMake is necessary as it is the control of the software compilation process. Our implementation uses the most recent version at the time, 3.7. The interactive option of CMake, ccmake is recommended and was used during the current installation. You can get the latest stable version from the Cmake Download Page [4]. A C++ compiler such as *g++* and *make* are necessary. Once the source code is downloaded and extracted, to install cmake, run the *bootstrap* script (here use the *–help* option to see the supported options, such as the *–prefix=custom install directory path* option), then *make* and *sudo make install* (or *make install* if a custom path was selected and no privileges are required). An additional library (*libncurses5-dev*) may be necessary for ccmake.

In summary:

```
1. $ sudo apt-get install libncurses5-dev
2. $ wget https://cmake.org/files/v3.7/cmake-3.7.1.tar.gz
3. $ tar -xvf cmake-3.7.1.tar.gz
4. $ cd cmake-3.7.1/
5. $ ./bootstrap
6. $ make
7. $ sudo make install
```

- Qt (version = 4.85)

Although it is possible to download and compile the source code from Qt download page [5], specifically from the Qt 4.85 version folder [6], it is preferable to install the corresponding package through the linux repository running the following command.

```
8. $ sudo apt-get install qt4-dev-tools
```

[4] https://cmake.org/cmake/resources/software.html
[5] http://download.qt.io/archive/
[6] http://download.qt.io/archive/qt/4.8/4.8.5/

- VTK (version = 7.0)

First download the source code from the VTK download page [7], decompress it and create a new folder (where the code will be built). Move to the new folder and run ccmake from build folder with argument the source folder. Press *c* to configure and then press *t* to display the advanced option and be sure that the following options have the values:

Module_vtkGUISupportQt: ON
Module_vtkGUISupportQtOpenGL:ON
Module_vtkGUISupportQtSQL: OFF
Module_vtkGUISupportQtWebkit: OFF
CMAKE_BUILD_TYPE: MinSizeRel
BUILD_SHARED_LIBS : ON
BUILD_DOCUMENTATION : OFF
BUILD_EXAMPLES : OFF
BUILD_TESTING : OFF

Then configure again by pressing *c* and generate the Makefile by pressing *g*. Then build the code by running *make* and finally *make install*. VTK will be installed normally in the folder: */home/(your username)/local/lib/cmake/vtk-7.0*, in my case: */home/laptop/local/lib/cmake/vtk-7.0*.

In summary:

```
$ wget http://www.vtk.org/files/release/7.0/VTK-7.0.0.tar.gz
$ tar -xvf VTK-7.0.0.tar.gz
$ mkdir VTK_Built
$ cd VTK_Built
$ ccmake ../VTK-7.0.0
Press c
Press t
Make sure the following options have the correct values:
    Module_vtkGUISupportQt: ON
    Module_vtkGUISupportQtOpenGL:ON
    Module_vtkGUISupportQtSQL: OFF
    Module_vtkGUISupportQtWebkit: OFF
    CMAKE_BUILD_TYPE: MinSizeRel
    BUILD_SHARED_LIBS : ON
    BUILD_DOCUMENTATION : OFF
    BUILD_EXAMPLES : OFF
    BUILD_TESTING : OFF
```

[7] http://www.vtk.org/download/

```
17. Press g
18. $ make
19. $ make install
```

- ITK (version = 4.10)

Download the source code from the ITK download page [8], decompress it and create a new folder (where the code will be built). Move to the new folder and run ccmake from build folder with argument the source folder. Press *c* to configure and then press *t* to display the advanced option and be sure that the following options have the values:

VTK_DIR : Folder where VTK is installed. In my case it was: /home/laptop/local/lib/cmake/vtk-7.0
Module_ITKReview : ON
Module_ITKVtkGlue : OFF
CMAKE_BUILD_TYPE: MinSizeRel
BUILD_SHARED_LIBS : ON
BUILD_DOCUMENTATION : OFF
BUILD_EXAMPLES : OFF
BUILD_TESTING : OFF

Then configure again by pressing *c* and generate the Makefile by pressing *g*. Then build the code by running *make* and finally *make install*. ITK will be installed normally in the folder: */home/(your username)/local/lib/cmake/ITK-4.10*, in my case: */home/laptop/local/lib/cmake/ITK-4.10*.

In summary:

```
20. $ wget http://downloads.sourceforge.net/project/itk/itk/4.10/
      InsightToolkit-4.10.1.tar.gz?r=https%3A%2F%2Fitk.org%2FITK%2
      Fresources%2Fsoftware.html&ts=1483297536&use_mirror=superb-dca2
21. $ tar -xvf InsightToolkit-4.10.1.tar.gz
22. $ mkdir ITK_Built
23. $ cd ITK_Built
24. $ ccmake ../InsightToolkit-4.10.1
25. Press c
26. Press t
27. Make sure the following options have the correct values:
    VTK_DIR : Folder where VTK is installed, for
    example: /home/laptop/local/lib/cmake/vtk-7.0
    Module_ITKReview : ON
```

[8] https://itk.org/ITK/resources/software.html

```
    Module_ITKVtkGlue : OFF
    CMAKE_BUILD_TYPE: MinSizeRel
    BUILD_SHARED_LIBS : ON
    BUILD_DOCUMENTATION : OFF
    BUILD_EXAMPLES : OFF
    BUILD_TESTING : OFF
28. Press g
29. $ make
30. $ make install
```

1.2 cpPlugins with plugins: FrontAlgorithms, cpPluginsLeFaucon

Professor Leonardo Flórez from Pontifica Universidad Javeriana is the main developper for cpPlugins and its plugins. Contact Leonardo [9] to have access to the source code for cpPlugins, FrontAlgorithms and cpPluginsLeFaucon and in case of having any difficulty related with cpPlugins, FrontAlgorithms and cpPluginsLeFaucon.

Once downloaded the source code for cpPlugins, FrontAlgorithms and cpPluginsLeFaucon proceed with the following steps for each project.

- cpPlugins

Decompress the source code and create a new folder (where the code will be built). Move to the new folder folder and run ccmake from the build folder with argument the source folder. Press *c* to configure and be sure that the following options have the values. You may have to press c every time an option is configured:

BUILD_PipelineEditor : ON
BUILD_EXAMPLES : ON
BUILD_MPRSeeds : ON
BUILD_MPRViewer : ON
BUILD_PipelineEditor : ON
BUILD_SIMPLE_TESTS : ON
CMAKE_BUILD_TYPE : Debug
CMAKE_INSTALL_PREFIX : /usr/local
ITK_DIR : Folder where ITK is installed, for example: /home/laptop/local/lib/cmake/ITK-4.10 QT_QMAKE_EXECUTABLE : /usr/bin/qmake
USE_QT4 : ON

[9] florez-l@javeriana.edu.co

VTK_DIR : Folder where VTK is installed, for example: /home/laptop/local/lib/cmake/vtk-7.0
cpPlugins_CONFIG_NUMBER_OF_FIL : 10
cpPlugins_CONFIG_PROCESS_DIMEN : 1;2;3
cpPlugins_CONFIG_VISUAL_DIMENS : 2;3
cpPlugins_Qt4_VTKWidget : QVTKWidget

Then configure again by pressing *c* and generate the Makefile by pressing *g*. Then build the code by running *make*.

In summary:

```
$ mkdir cpPlugins_Built
$ cd cpPlugins_Built
$ ccmake folder where the source of cpPlugins is.
Press c
Make sure the following options have the correct values:
    BUILD_PipelineEditor : ON
    BUILD_EXAMPLES : ON
    BUILD_MPRSeeds : ON
    BUILD_MPRViewer : ON
    BUILD_PipelineEditor : ON
    BUILD_SIMPLE_TESTS : ON
    CMAKE_BUILD_TYPE : Debug
    CMAKE_INSTALL_PREFIX : /usr/local
    ITK_DIR : Folder where ITK is installed, for
     example: /home/laptop/local/lib/cmake/ITK-4.10
    QT_QMAKE_EXECUTABLE : /usr/bin/qmake
    USE_QT4 : ON
    VTK_DIR : Folder where VTK is installed, for
     example: /home/laptop/local/lib/cmake/vtk-7.0
    cpPlugins_CONFIG_NUMBER_OF_FIL : 10
    cpPlugins_CONFIG_PROCESS_DIMEN : 1;2;3
    cpPlugins_CONFIG_VISUAL_DIMENS : 2;3
    cpPlugins_Qt4_VTKWidget : QVTKWidget
Press g
$ make
```

- FrontAlgorithms

Decompress the source code and create a new folder (where the code will be built). Move to the new folder folder and run ccmake from the build folder with argument the source folder. Press *c* to configure and be sure that the following options have the

values. You may have to press c every time an option is configured:

BUILD_EXAMPLES : OFF
BUILD_EXPERIMENTS : OFF
BUILD_ExperimentationPlugins : OFF
CMAKE_BUILD_TYPE : Debug
CMAKE_INSTALL_PREFIX : /usr/local
ITK_DIR : Folder where ITK is installed, for example: /home/laptop/local/lib/cmake/ITK-4.10
USE_cpPlugins : ON
VTK_DIR : Folder where VTK is installed, for example: /home/laptop/local/lib/cmake/vtk-7.00
cpPlugins_BaseLibraries : cpPlugins;cpExtensions;cpPluginsDataObjects;cpBaseQtApplication
cpPlugins_DIR : Folder where cpPlugins is built.

Then configure again by pressing *c* and generate the Makefile by pressing *g*. Then build the code by running *make*.

In summary:

```
$ mkdir FrontAlgorithms_Built
$ cd FrontAlgorithms_Built
$ ccmake folder where the source of FrontAlgorithms is.
Press c
Make sure the following options have the correct values:
    BUILD_EXAMPLES : OFF
    BUILD_EXPERIMENTS : OFF
    BUILD_ExperimentationPlugins : OFF
    CMAKE_BUILD_TYPE : Debug
    CMAKE_INSTALL_PREFIX : /usr/local
    ITK_DIR : Folder where ITK is installed, for
     example: /home/laptop/local/lib/cmake/ITK-4.10
    USE_cpPlugins : ON
    VTK_DIR : Folder where VTK is installed, for
     example: /home/laptop/local/lib/cmake/vtk-7.00
    cpPlugins_BaseLibraries : cpPlugins;cpExtensions;cpPluginsDataObjects;
     cpBaseQtApplication
    cpPlugins_DIR : Folder where cpPlugins is built.
Press g
$ make
```

- cpPluginsLeFaucon

Decompress the source code and create a new folder (where the code will be built). Move to the new folder folder and run ccmake from the build folder with argument the source folder. Press *c* to configure and be sure that the following options have the values. You may have to press c every time an option is configured:

CMAKE_BUILD_TYPE : Debug
CMAKE_INSTALL_PREFIX : /usr/local
ITK_DIR : Folder where ITK is installed, for example: /home/laptop/local/lib/cmake/ITK-4.10
USE_cpPlugins : ON
VTK_DIR : Folder where VTK is installed, for example: /home/laptop/local/lib/cmake/vtk-7.00
cpPlugins_BaseLibraries : cpPlugins;cpExtensions;cpPluginsDataObjects;cpBaseQtApplication
cpPlugins_DIR : Folder where cpPlugins is built.

Then configure again by pressing *c* and generate the Makefile by pressing *g*. Then build the code by running *make*.

In summary:

```
$ mkdir cpPluginsLeFaucon_Built
$ cd cpPluginsLeFaucon_Built
$ ccmake folder where the source of cpPluginsLeFaucon is.
Press c
Make sure the following options have the correct values:
    CMAKE_BUILD_TYPE : Debug
    CMAKE_INSTALL_PREFIX : /usr/local
    ITK_DIR : Folder where ITK is installed, for
     example: /home/laptop/local/lib/cmake/ITK-4.10
    USE_cpPlugins : ON
    VTK_DIR : Folder where VTK is installed, for
     example: /home/laptop/local/lib/cmake/vtk-7.00
    cpPlugins_BaseLibraries : cpPlugins;cpExtensions;cpPluginsDataObjects;
     cpBaseQtApplication
    cpPlugins_DIR : Folder where cpPlugins is built.
Press g
$ make
```

1.3 BBTK and CreaTools

Researcher Eduardo Dávila from the CREATIS Laboratory (Lyon, France) is the main developper for BBTK and CreaTools. Contact Eduardo [10] if there is any difficulty related with BBTK and CreaTools.

- CreaTools

Simply download the CreaTools installation script at the CreaTools download page [11] and run it with administration privileges. The script will install the libraries necessary for CreaTools and the different tools in the system.

In summary:

```
45. $ wget http://www.creatis.insa-lyon.fr/software/public/creatools/
      creaTools/Install-Creatools-Bin-Release.sh
46. $ source ./Install-Creatools-Bin-Release.sh
```

- BBTK part from our project

Download our project source. In order to do it contact Sergio[12]. Decompress it. The project has a folder structure Images, Results, script.

In summary:

```
47. Download the project source
48. Decompress the project source
```

1.4 Final adjustments

In order to connect cpPlugins with our project a few more steps are necessary. First go to the script folder of our project, then create a link to the executable for cpPlugins with the command *ls -s TARGET LINK_NAME*and finally create a file called *cpPlugins_PATHS* which will contain the paths for the built versions of cpPlugins, FrontAlgorithms and cpPluginsLeFaucon.

In summary:

[10] Eduardo.Davila@creatis.insa-lyon.fr
[11] https://www.creatis.insa-lyon.fr/site/en/CreatoolsDownload.html
[12] sd.hernandez204@uniandes.edu.co

49. $ **cd** OurProject/script
50. $ ln -s pathToBuiltCpPlugins/cpPlugins_plugins_ExecutePipeline
    cpPlugins_plugins_ExcecutePipeline
51. Create a plain text file called cpPlugins_PATHS with:
    path to our project/script (current path)
    path to the folder of the built version of FrontAlgorithms
    path to the folder of the built version of cpPlugins
    path to the folder of the built version of cpPluginsLeFaucon

Once all theses steps have been followed, our project should be ready to be executed.