Rotation – MRI Images Registration

Registration of MRI Images using the SimpleElastix module compiled for SimpleITK for use in Python

Wayan Fontaine-Seiler
Department of Physics, Georgetown University,
1st Year Ph.D. Student

June – July 2020

Contents

Introduction

This documentation summarizes the research carried out during the rotation with Edward Van Keuren and the first part will also serve as a user's installation manual and brief introduction to the SimpleElastix[3] tool for whoever might be interested.

This documentation in the first part is intended to complement and bring additional details to the already existing documentation online[1][4].

[1]`https://simpleelastix.readthedocs.io/`

Compiling the module for the different platforms

1.1 Introduction

The extension has to be compiled from source as the standard SITK module found in most Python distributions (such as Anaconda) does not come with the SimpleElastix addition. The use of the standard SITK extension will return errors when trying to access those specific functions.

The compiling of the extension might be a difficult task and can take, depending on the hardware used, several hours to complete.

1.2 Compiling for a Linux/GNU distribution

Compiling for a Linux/GNU distribution is the easiest and only requires common packages to be successful. These instructions assume that you run a system based on the x86_64 (also known as x64 or amd64) architecture.

This guide also assumes a basic knowledge of the terminal and of the most common terminal commands as well as basic knowledge of you distribution (software/package install through the terminal mainly). Nonetheless, most steps will be detailed and explained.

This guide should be valid for most Linux/GNU distributions but has only been tested with Debian and Arch Linux.

1.2.1 Setting up a virtual Python environment

Installation This is not a required step but is one that we would heavily recommend, especially if you are new to Python and Python environments. Setting up a virtual environment will protect your system environment from misuses and give you greater control over the environment you will use for this module. This will also enable you to install packages without root/sudo privileges if you do not have administrator privileges on your machine.

We will describe here how to set up an environment using Anaconda[1] but you can use another way if you are already familiar with other tools. Bear in mind that the compiling instructions have been written with this distribution of Python in mind and adjustments might be required if you use another one.

The first step is to download the correct version of the Anaconda distribution. The version you should select is the Python 3.7 (or later) 64-bit installer for the x86 architecture. The follow the instructions at `https://docs.anaconda.com/anaconda/install/linux/`[2] (installing the dependencies and Anaconda) to complete the installation process.

At that point, We suggest getting a little familiar with Anaconda using the online documentation[2] especially with the `conda activate / deactivate`, `conda install` and `conda create` commands.

Creating the virtual environment To create virtual environment, run the command: `conda create -n envname` with envname the name you want for your environment. We will use here “mrireg” (for MRI registration).

You can specify a version of Python if you so choose by using the argument `python=3.X`, but we suggest leaving it at default. An extensive documentation is available online for the creation and management of environments[3].

Activate the environment using `conda activate mrireg` (or the name you have chosen) and install the following required or recommended packages (NumPy, scipy, matplotlib, PILLOW, spyder) by running `conda install numpy scipy matplotlib PILLOW spyder` (you can omit `spyder` if you want to use another IDE) and type “y” and enter to proceed. This will install some necessary modules as well as useful modules.

Install the pydicom package with `conda install -c conda-forge pydicom`. This will install a module that handles DICOM images.

Install OS with `conda install -c jmcmurray os`. This will install os, a tool that allows us to test OS functions, test system commands on the current platform.

You can also install all these modules using PIP if you so choose, but you will first need to install PIP with `conda install pip` and `pip install` the necessary packages but installing through `conda install` is recommended for Anaconda.

[1]Available at `https://www.anaconda.com/products/individual`

[2]`https://docs.anaconda.com/`

[3]`https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html`

1.2.2 Compiling the module

This section borrows heavily on the SimpleElastix documentation[4] but some important steps to ensure successful compiling and installation have been added.

Most of the steps described below make use of the terminal, but you can perform some of them with GUI tools is you so choose. Make sure you have enough disk space available. Compiling the project could take up to 15GB of disk space.

First, install the required dependencies for the compilation part: `cmake git build-essential`. You will also need some of the following dependencies for the wrapping part: `python python-dev monodevelop r-base r-base-dev ruby ruby-dev tcl tcl-dev tk tk-dev`. Bear in mind that these are the names of these packages for Debian-based and Ubuntu-based distribution, and they might have other names in other distributions. Here, we can restrict ourselves to `python` and `python-dev` since we only need the PYTHON wrapper/installer (`sudo apt-get install python python-dev` in the case of a Debian or Ubuntu-based distribution for example). You should also make sure that the package `swig` is installed in you distribution as it will be necessary when generating the wrapper amd might not come with the build tools of your distribution.

Open a terminal and navigate to you home folder. Create a directory named `/SimpleElastix` by running `mkdir SimpleElastix`. Move to that directory (`cd SimpleElastix`) and create a directory named `/build` with `mkdir build`. In `SimpleElastix`, git clone the SIMPLEELASTIX project (`git clone https://github.com/SuperElastix/SimpleElastix`) and move to the build directory created previously (`cd build`).

The next step is a little unorthodox, as it involves compiling the program with the virtual environment active, but gave us great results regarding the compilation and installation process. Activate your virtual PYTHON environment, if it is not already active, and run the following commands:

- `cmake ../SimpleElastix/SuperBuild` This will create a makefile for the compilation part. If you need to adjust some parameters, you could always install and call `cmake-gui` and make the necessary modifications.
- `make -jN` with N the number of threads you intend to use for the compiling process, a higher number will lead to more parallelization and lower compilation time. To choose this number, we suggest considering the number of cores of your CPU, its hyperthreading capabilities, and the amount of RAM. If your CPU is hyperthreading capable, you can go up to twice the number of physical cores CPU and, if not, the number of physical cores of your CPU. RAM might also be a bottleneck. A conservative approach to compiling and the online documentation recommend considering 4GB per thread, but in that case, we noticed that the RAM consumption never exceeds 3GB per thread on average[5]. In practice, look

[4]https://simpleelastix.readthedocs.io/

[5]If you run out of physical RAM and do not have a swap partition or swapfile, your system might

at the number of parallel threads your CPU can support and consider the integer floor(RAM in GB/3) and take the lowest of both integer[6][7].

Depending on you machine, the compiling process could take from 10 minutes to a few hours.

1.2.3 Installation process

Navigate to `~/SimpleElastix/build/SimpleITK/Wrapping/Python` (`cd ~/SimpleElastix/build/SimpleITK/Wrapping/Python`) and copy the file `_SimpleITK.so` in the `/Packaging` directory (`cp _SimpleITK.so Packaging`). You can now run `python Packaging/setup.py install`.

The module should now be installed and ready to use. You can test if it is functioning correctly by running a file containing this simple line: `import SimpleITK as sitk`. No errors should appear. If some errors appear, see section 1.5 – Troubleshooting and Important notes.

1.3 Compiling for Microsoft Windows

We will assume, in this section, that you use Windows 10, but these instructions should be valid for previous releases of this operating system. Like in section 1.2 – Compiling for a Linux/GNU distribution, you will need a large disk space to compile this project (around 15GB). You will need to add to that the Visual Studio development tools (a few GB).

This guide borrows heavily from the guide available online[8] but adds essential details to ensure a successful compiling and installation.

Compiling on Windows is not as straightforward as on Linux/GNU and allows for less flexibility.

In this section, we suggest setting up the Python virtual environment after the compiling of the program.

crash. If your system starts swapping, it might become slow and unresponsive, thus increasing the compiling time.

[6]Example: In our case, the compiling was run on an 8 core, 16 threads machine with 32GB of RAM. We therefore ran `make -j10`.

[7]If you get an error during the compiling process, try a lower value for `-jX` like `make -j4` as higher values might create instabilities in that case

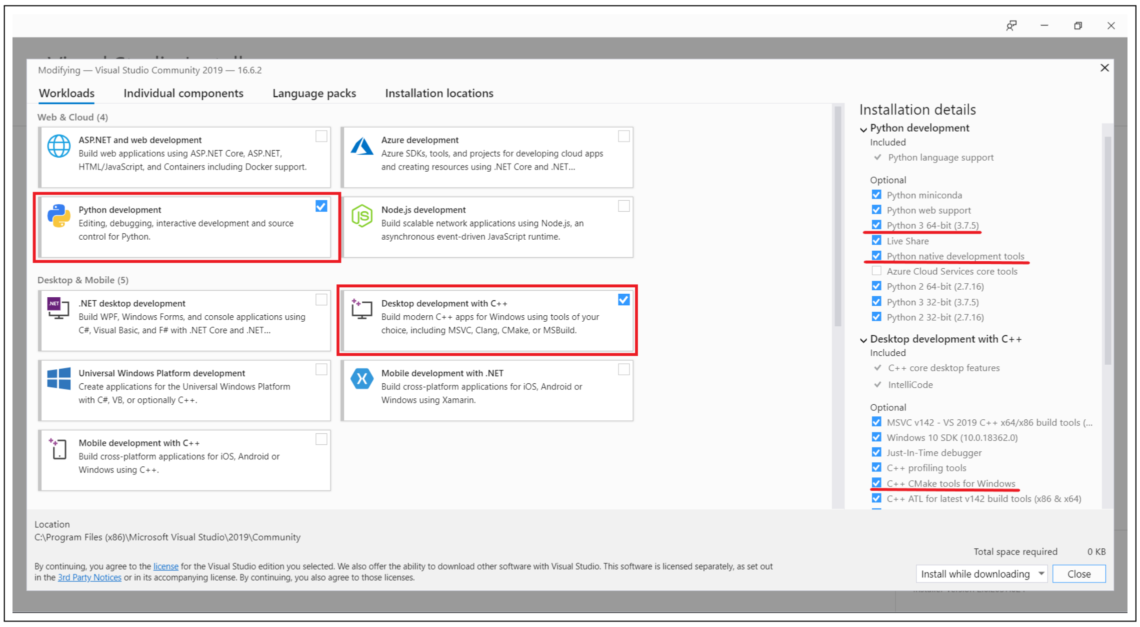
[8]`https://simpleelastix.readthedocs.io/GettingStarted.html`

1.3.1 Required software and installation

In order to compile the program, you will first need to download and install some programs:

- CMake GUI[9]: This tool will create a makefile for the Visual Studio compiler
- Git[10]: This tool will allow you to git clone the project on Windows with a UNIX-style command line tool.
- Visual Studio 2019 Community Edition[11]: Provides an x64 compiler and a Python development environment

Regarding Visual Studio 2019, you will need to install additional tools and features to be able to compile the module. To install those tools, open Visual Studio 2019 and click on "Continue without code". Navigate to "Tools"≫"Get Tools and Features". Tick "Python Development" and "Desktop development with C++" and make sure that **Python 3 64-bit**, **Python native development tools** and **C++ CMake tools** are selected (see fig.1.3.1).


Modifying — Visual Studio Community 2019 — 16.6.2

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

- [ ] ASP.NET and web development — Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.
- [ ] Azure development — Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET...
- [✓] Python development — Editing, debugging, interactive development and source control for Python.
- [ ] Node.js development — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- [ ] .NET desktop development — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...
- [✓] Desktop development with C++ — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- [ ] Universal Windows Platform development — Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
- [ ] Mobile development with .NET — Build cross-platform applications for iOS, Android or Windows using Xamarin.
- [ ] Mobile development with C++ — Build cross-platform applications for iOS, Android or Windows using C++.

Installation details

Python development

Included
- ✓ Python language support

Optional
- [✓] Python miniconda
- [✓] Python web support
- [✓] Python 3 64-bit (3.7.5)
- [✓] Live Share
- [✓] Python native development tools
- [ ] Azure Cloud Services core tools
- [✓] Python 2 64-bit (2.7.16)
- [✓] Python 3 32-bit (3.7.5)
- [✓] Python 2 32-bit (2.7.16)

Desktop development with C++

Included
- ✓ C++ core desktop features
- ✓ IntelliCode

Optional
- [✓] MSVC v142 - VS 2019 C++ x64/x86 build tools (...
- [✓] Windows 10 SDK (10.0.18362.0)
- [✓] Just-In-Time debugger
- [✓] C++ profiling tools
- [✓] C++ CMake tools for Windows
- [✓] C++ ATL for latest v142 build tools (x86 & x64)

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 0 KB

Figure 1.3.1: Screenshot of Visual Studio installation tool with the different options to install circled and underlined

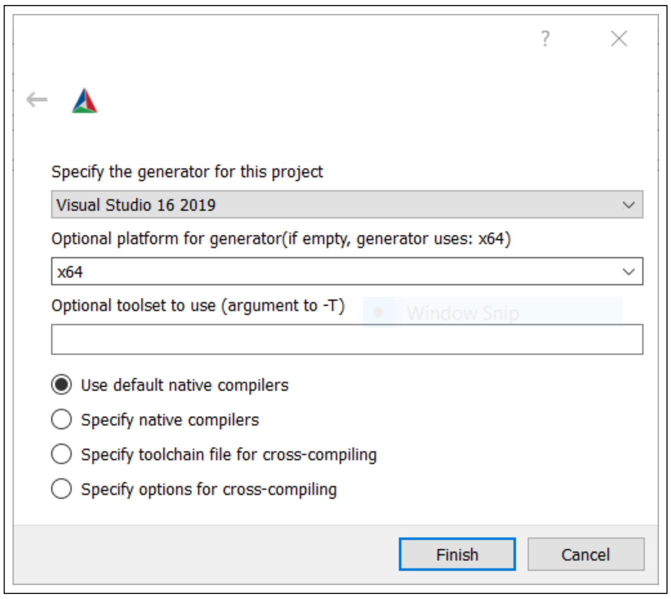
Install. This will install a compiler as well as some necessary libraries in C++ and Python. Pay attention to the version of Python installed as it will be required later when setting the virtual environment in subsection 1.3.3 – Setting up the virtual environment and installing the module.

[9]Available at `https://cmake.org/download/`
[10]Available at `https://git-scm.com/downloads`
[11]Available at `https://visualstudio.microsoft.com/vs/`

1.3.2 Compiling the module

Git cloning the project Open GIT BASH and navigate to the root of your `C:` drive (VISUAL CODE compiler has trouble with paths longer than 50 characters). Create a directory named `\SimpleElastix` (`mkdir SimpleElastix`), move in that directory (`cd SimpleElastix`), git clone the project (`git clone https://github.com/kaspermarstal/SimpleElastix` – **WARNING:** This repository is different than the LINUX/GNU one). Create a `\build` directory (`mkdir build`).

Cmake Open CMAKE GUI and point the source to `SimpleElastix\SuperBuild` and the build directory to the `build` directory you just created. Click on "configure" select the values as shown in figure 1.3.2 for the different fields.


Specify the generator for this project
Visual Studio 16 2019
Optional platform for generator(if empty, generator uses: x64)
x64
Optional toolset to use (argument to -T)

● Use default native compilers
○ Specify native compilers
○ Specify toolchain file for cross-compiling
○ Specify options for cross-compiling

Figure 1.3.2: Screenshot of the configuration window of CMAKE GUI with the correct values for the different fields

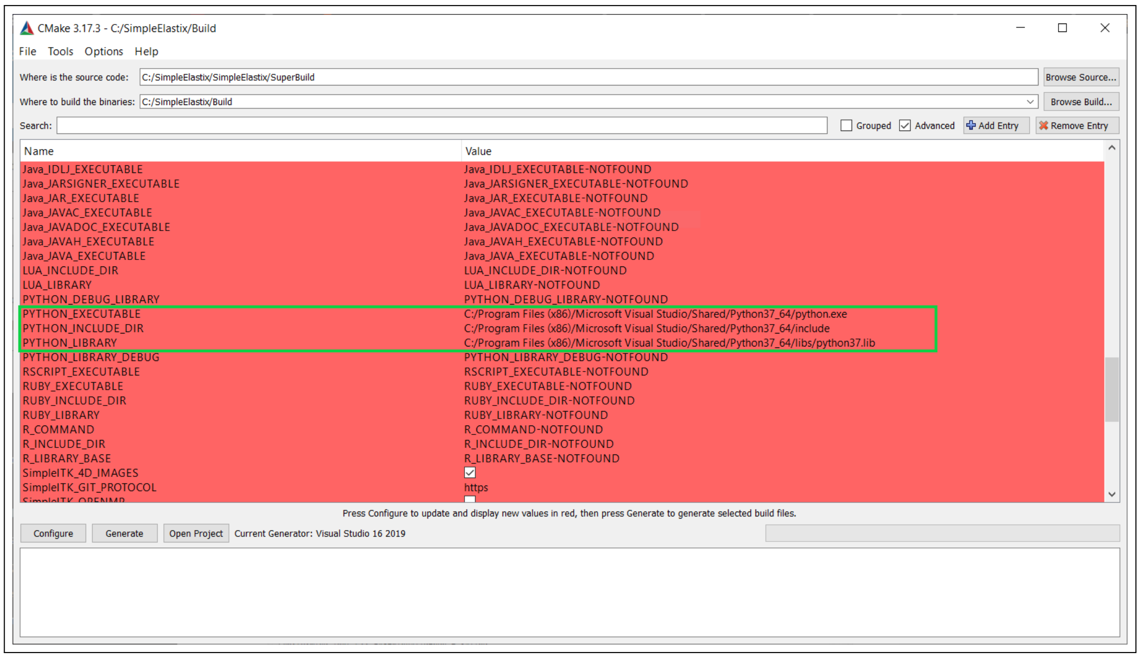
Tick the "Advanced" box and look for PYTHON_EXECUTABLE, PYTHON_INCLUDED_DIR, PYTHON_LIBRARY and make sure they are the same version of PYTHON you noted previously in 1.3.1[12] (See fig. 1.3.3).

After checking that everything is in order, click on "Generate" and wait for the "Generating done".

Compiling the module itself using x64 Native Tools Command Prompt for VS 2019 There are several ways of compiling the module, but the easiest by far is through the terminal, as it involves only one command.

Open **x64 Native Tools Command Prompt for VS 2019** and navigate to `C:\SimpleElastix\build` (using the `cd` command and `cd ..` to go up one directory

[12]Here, for instance, python37_64 means PYTHON version 3.7 for an x64 architecture.



Where is the source code: C:/SimpleElastix/SimpleElastix/SuperBuild

Where to build the binaries: C:/SimpleElastix/Build

Search: ☐ Grouped ✓ Advanced

Name	Value
Java_IDLJ_EXECUTABLE	Java_IDLJ_EXECUTABLE-NOTFOUND
Java_JARSIGNER_EXECUTABLE	Java_JARSIGNER_EXECUTABLE-NOTFOUND
Java_JAR_EXECUTABLE	Java_JAR_EXECUTABLE-NOTFOUND
Java_JAVAC_EXECUTABLE	Java_JAVAC_EXECUTABLE-NOTFOUND
Java_JAVADOC_EXECUTABLE	Java_JAVADOC_EXECUTABLE-NOTFOUND
Java_JAVAH_EXECUTABLE	Java_JAVAH_EXECUTABLE-NOTFOUND
Java_JAVA_EXECUTABLE	Java_JAVA_EXECUTABLE-NOTFOUND
LUA_INCLUDE_DIR	LUA_INCLUDE_DIR-NOTFOUND
LUA_LIBRARY	LUA_LIBRARY-NOTFOUND
PYTHON_DEBUG_LIBRARY	PYTHON_DEBUG_LIBRARY-NOTFOUND
PYTHON_EXECUTABLE	C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/python.exe
PYTHON_INCLUDE_DIR	C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/include
PYTHON_LIBRARY	C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/libs/python37.lib
PYTHON_LIBRARY_DEBUG	PYTHON_LIBRARY_DEBUG-NOTFOUND
RSCRIPT_EXECUTABLE	RSCRIPT_EXECUTABLE-NOTFOUND
RUBY_EXECUTABLE	RUBY_EXECUTABLE-NOTFOUND
RUBY_INCLUDE_DIR	RUBY_INCLUDE_DIR-NOTFOUND
RUBY_LIBRARY	RUBY_LIBRARY-NOTFOUND
R_COMMAND	R_COMMAND-NOTFOUND
R_INCLUDE_DIR	R_INCLUDE_DIR-NOTFOUND
R_LIBRARY_BASE	R_LIBRARY_BASE-NOTFOUND
SimpleITK_4D_IMAGES	✓
SimpleITK_GIT_PROTOCOL	https

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 16 2019

Figure 1.3.3: Screenshot of CMake GUI with the relevant fields circled in green – They should read the same Python version as installed in Visual Code 2019.

in the tree). Run `msbuild ALL_BUILD.vcxproj /p:Configuration=Release`. Depending on your hardware, the compiling time may vary from 20 minutes to several hours[13]. You might get a few compilation warnings during the compiling process and at the end of the process but this should not be a problem for us ands should not affect the Python module.

Preparing the module for installation To prepare the module for installation, navigate, using the file explorer to `C:\SimpleElastix\build\SimpleITK-build\Wrapping\Python` and copy `_SimpleITK.pyd` to the `\Packaging` folder.

1.3.3 Setting up the virtual environment and installing the module

Like in subsection 1.2.1 – Setting up a virtual Python environment, we will need to install Anaconda for Windows[14]. Make sure to download the Python 3.7 64-Bit (or later) version and follow the sames steps as in subsection 1.2.1 – Setting up a virtual Python environment to set up the virtual environment. The only difference are:

- In the terminal: on Windows, launch "Anaconda Prompt" as an administrator

[13]In our case, on a 6 core, 12 threads machine with 16GB of RAM, the compiling process took around 1h45.

[14]Available at `https://www.anaconda.com/products/individual`

- When creating the PYTHON environment, make sure to use the correct version you noted from 1.3.1 and 1.3.2 using the command `conda create -n envname python=3.X`[15].

The rest of the steps are identical.

To install the module, navigate to C:\SimpleElastix\build\SimpleITK-build\Wrapping\Python\Packaging and run `python setup.py install` with your target environment active. As with LINUX/GNU, there should be no errors when running a simple file containing the following line: `import SimpleITK as sitk`.

1.4 Compiling for macOS / Mac OS X

As macOS / Mac OS X is based on BSD, an operating system with a UNIX kernel, the instructions in section 1.2 – Compiling for a LINUX/GNU distribution will also work for this OS and have been tested on macOS 10.13 High Sierra[16][17].

The only difference is in the installation of CMAKE. On macOS, you will need to compile CMAKE from source as the provided binairies did not work in our case.

To compile and install CMAKE, first download the UNIX source[18] and unpack it and move to the directory. You can follow the official installation guide[1] by running the following commands in the terminal:

- `.\bootstrap` This command will create the necessary makefile and compilation files for the next step of the compiling.
- `make` This step will compile the program.
- `sudo make install` This will install the compiled program in your system and it will require administrator privileges.

After installing CMAKE, you can follow the same steps as in section 1.2 – Compiling for a LINUX/GNU distribution.

[15]In our case, we would run `conda create -n mrireg python=3.7` since the developer tools where for PYTHON 3.7 in VISUAL STUDIO 2019

[16]At the time of writing, the latest public version of macOS was macOS 10.15 Catalina, the module was therefore compiled on an older version of macOS.

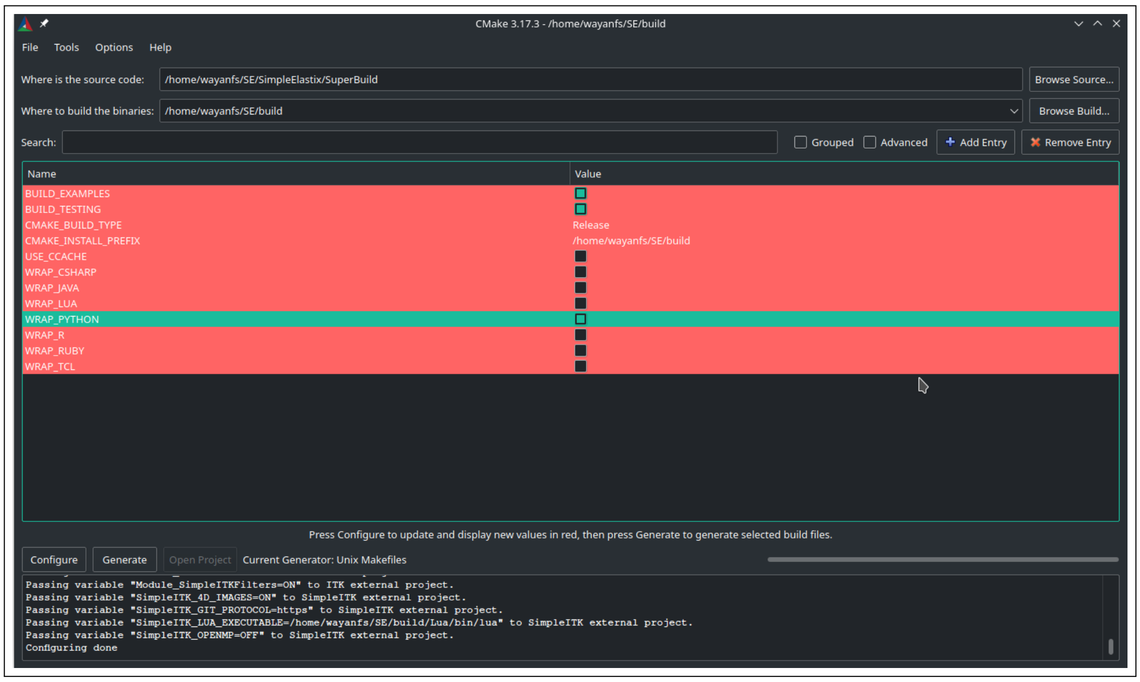
[17]As of the time of writing this documentation, APPLE just started its transition from an INTEL based x86_64 / x64 / amd64 architecture to its own ARM-based APPLE SILICON architecture. This might render this guide completely obsolete for macOS in the future.

[18]Available at `https://cmake.org/download/`

1.5 Troubleshooting and Important notes

1.5.1 A few notes and troubleshooting steps

- During the compiling process in section 1.2 – Compiling for a LINUX/GNU distribution, if you encounter an error from `make`, try running `cmake-gui` (while still in your PYTHON virtual environment), point the build and source directory to the adequate directories (see paragraph 1.3.2 – CMAKE in section 1.3 – Compiling for MICROSOFT WINDOWS), generate and use the native compilers when prompted, untick all wrappings except the PYTHON one (see fig.1.5.1) .



Where is the source code: /home/wayanfs/SE/SimpleElastix/SuperBuild

Where to build the binaries: /home/wayanfs/SE/build

Search:

Name	Value
BUILD_EXAMPLES	🟩
BUILD_TESTING	🟩
CMAKE_BUILD_TYPE	Release
CMAKE_INSTALL_PREFIX	/home/wayanfs/SE/build
USE_CCACHE	■
WRAP_CSHARP	■
WRAP_JAVA	■
WRAP_LUA	■
WRAP_PYTHON	☐
WRAP_R	■
WRAP_RUBY	■
WRAP_TCL	■

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

```
Passing variable "Module_SimpleITKFilters=ON" to ITK external project.
Passing variable "SimpleITK_4D_IMAGES=ON" to SimpleITK external project.
Passing variable "SimpleITK_GIT_PROTOCOL=https" to SimpleITK external project.
Passing variable "SimpleITK_LUA_EXECUTABLE=/home/wayanfs/SE/build/Lua/bin/lua" to SimpleITK external project.
Passing variable "SimpleITK_OPENMP=OFF" to SimpleITK external project.
Configuring done
```

Figure 1.5.1: Screenshot of CMAKE GUI with the only wrapper necessary for our use case.

  Delete build files and recompile, this should solve the problem. For the same problem in macOS, i suggest the same procedure but the regular CMAKE GUI binary from the website will not work. You will need to install MACPORTS[19] first and run `sudo port install cmake +gui` to install CMAKE GUI.

- The module seems to be very sensitive to the PYTHON version it is running in. For instance, the module compiled against the Python 3.7 libraries will not work with Python 3.8. This is why we recommend the unorthodox approach of compiling with the virtual environment active in 1.2.2 as this ensures that the proper version of Python will be used for the compiling process[20]. This is also why we recommend setting up the virtual environment after the compiling process on WINDOWS.

[19]Available at `https://www.macports.org/install.php`

[20]Linux distributions come with a wide variety of PYTHON versions and the procedure described in 1.2 ensures reproducible compiling and consistent usage experience and behavior across distributions and platforms.

- On Linux/GNU, you can easily compile the module against any version of Python using virtual environments if you need to use a specific version of Python. While possible on Windows, it is significantly more complex as you will need to locate the Python executable and the required libraries manually.
- If you run into issues during the compiling, the installation, or post-installation, we suggest looking at the following websites: `https://docs.conda.io/projects/conda/en/latest/index.html`, `https://simpleelastix.readthedocs.io/`. Unfortunately, not a lot of documentation exists for this specific module, and they generally lack important details, hence this installation guide.
- If you get errors stating that the module does not exist, it might be due to mismatched versions of Python use for compiling and for the environment. As we had the same issues, we suggest checking back everything and recompiling, ensuring that the versions match.

1.5.2 Advanced compiling on Linux/GNU, Windows and macOS

Bibliography

[1] CMake *Installation Guide.* [Online; accessed 11-July-2020]. URL: `https://cmake.org/install/`.

[2] *Conda Documentation – Anaconda.* [Online; accessed 10-July-2020]. URL: `https://docs.conda.io/projects/conda/en/latest/index.html`.

[3] Kasper Marstal et al. "SimpleElastix: A User-Friendly, Multi-lingual Library for Medical Image Registration". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).* IEEE, June 2016. DOI: `10.1109/cvprw.2016.78`. URL: `https://doi.org/10.1109/cvprw.2016.78`.

[4] *SimpleElastix Documentation.* [Online; accessed 10-July-2020]. URL: `https://simpleelastix.readthedocs.io/`.

List of Figures

List of Tables