AngioTK practical work

from MRA to vascular 3D model

François Der Hovsepian

System requirements

The only requirement is to have docker installed on your system. The docker daemon should be up and running before you start this workshop.

Moreover, it is highly recommended to make sure your system has enough RAM (4GB or more) as some applications (surface extraction and center lines computing) are rather memory-hungry.

NOTE The graphical interfaces have only been tested on Linux hosts, and may either require modifications to work on other host systems, or even not work at all.

NOTE The instructions presented here have been successfully tested under Linux distributions based hosts (CENTOS 7, Ubuntu 14.04 & 16.04). For each system, a fresh install was performed and docker was installed afterwards.

Preparing the files and launching docker

The `tp/` directory

All the files needed for this workshop are provided in a directory named `tp/`. You should put it in your home directory as the docker running script is tailored to work this way. Otherwise, you will need to edit the script and adapt the following instructions.

NOTE All the commands provided here have to be typed in a console.

Starting docker

You can now start a docker container running the AngioTK image. For your convenience, a script featuring the docker run command and all the required arguments is provided, so you only need to execute it:

```
cd ~/tp
. docker_run_command.sh
```

TIP The second line starts with a simple dot `.` which is equivalent to the `source` command. It will be used to call scripts throughout this document.

You should now see the AngioTK environment:

```
# Feel++ applications -*- mode: markdown -*-

This image provides the Feel++ applications in
 . advection
 . fluid mechanics
 . solid mechanics
 . fluid-structure interaction
 . thermodynamics
 . thermoelectric

The testcases are in $HOME/Testcases/ and the results will be stored in
/feel.

Follow the steps described in

https://book.feelpp.org/content/en/03-modeling/

to use our toolboxes.
feelpp@8d5983bfd27a:~$
```

The docker container is ready and you can now start testing AngioTK.

NOTE You are now in a isolated environment (read more about docker container for a better understanding). However, thanks to some of the options provided in our `docker_run_command.sh` script, you have access to the `tp/` directory in `home/feelpp/tp` (or its shortcut `~/tp`).

Using AngioTK applications

AngioTK features several applications, which are supposed to be chained together and used in a pipeline fashion: (at least part of) the output of an application is (at least part of) the input of the next application in the pipeline.

Each application is launched in command line, which implies typing the command name as well as a few arguments. For example, to filter an MRA using `RORPO`, the complete command looks like this:

```
RORPO_multiscale_usage <input.nii> <output.nii> <initial path length> <path length
growth factor> <total number of path lengths> --core <number of core> --verbose
```

Other commands are even longer… As you can imagine, typing every command entirely can be tiresome at best. For this reason, we provide a set of scripts to call each application. For example, to run `RORPO`, simply type:

```
cd ~/tp
. 1_rorpo.sh
```

Calling the script saves you typing its content, which is:

```
(time RORPO_multiscale_usage /home/feelpp/tp/0_iso/pcp_iso.nii
/home/feelpp/tp/1_rorpo/pcp_25_1.34_7_4.nii 25 1.34 7 --core 4 --verbose) 2>&1 | tee
/home/feelpp/tp/log/1_rorpo.log
```

As you can see, the command is quite long, and is made even longer by adding two commands - `time` and `tee` - which are used to report execution times and log console output respectively. You can thus read log files in `~/tp/log/`, and any log ends with the execution times.

NOTE The tradeoff of using these scripts is that all of the arguments (including input and output path) are already set. We tried to make the paths predictable, with directory names matching the pipeline flow: see how `RORPO` writes its output to `~/tp/1_rorpo`. Feel free to change any argument by editing the scripts !

Complete list of applications

AngioTK applications are listed here in the pipeline order, with the name of the corresponding script:

- `vmtkimagewriter` (initial `.mha` to `.nii` conversion): `0_mhatonii.sh`
- `RORPO` (3D image filtering): `1_rorpo.sh`
- `vmtkimagewriter` (`.nii` to `.mha` conversion): `1b_niitomha.sh`
- `angiotk_meshing_surfacefromimage` (segmentation / surface extraction): `2_sfi.sh`
- `angiotk_meshing_centerlinesmanagergui` (manual center lines definition using a GUI): `gui.sh`
- `angiotk_meshing_centerlines` (center lines computing): `3_cl.sh`
- `angiotk_meshing_centerlinesmanager` (center lines merging): `4_cl.sh`
- `angiotk_meshing_imagefromcenterlines` (3D image creation from center lines): `5_ifcl.sh`
- `angiotk_meshing_surface` (second surface extraction): `6_ssfi.sh`
- `angiotk_meshing_remeshstl` (surface processing): `7_sp.sh`
- `angiotk_meshing_volumefromstlandcenterlines` (volume processing): `8_vp.sh`

Please refer to AngioTK's documentation for detailed information about these application.

The superscript(s)

To make it even easier to test the whole pipeline, we provide 3 "superscripts":

- `superscript.sh`: start the whole pipeline.
- `no_rorpo_super_script.sh`: skip the first steps (`RORPO` and image conversions) and start directly with surface extraction
- `no_ifcl_super_script.sh`: skip more steps (up to and including image creation from center lines)

and start with the second surface extraction.

The idea is to skip the most time-consuming steps - image filtering and center lines computing - whenever their results are satisfying.

NOTE This can be combined with the `force-rebuild` option available in most of the configuration files (which are in `tp/cfg/`), which allow to avoid computation if the target output already exists.

To get total execution times and entire output logging, you should use the following scripts instead, respectively:

- `start.sh`
- `skip_rorpo_start.sh`
- `skip_ifcl_start.sh`

Testing the pipeline

Complete test

To test the whole pipeline at ounce, while in the docker container, simply run the complete superscript:

```
cd ~/tp
. start.sh
```

Using the provided files, you shouldn't run into any error. That's the easiest way to make sure everything runs as expected, so it is a good idea to do it ounce before trying anything else.

Using your own center line files

If you created your own center line file (using the gui editor) and wish to use it instead of the provided `pointpair.data`, make sure to edit the center lines computing script (`3_cl.sh`) to adjust the point-pair file path. Then, to check your progress, run `skip_rorpo_start.sh` to start the pipeline directly at the center line computing step and avoid running all the previous steps again and again.

Changing other parameters

Feel free to play with the various parameters - while some are in the scripts themselves, others are in the corresponding configuration files in `tp/cfg/`. Refer to AngioTK's documentation for more information.