24783 Advanced Engineering Computation: Problem Set 3

(*) In the following instruction (and in all of the course materials), substitute your Andrew ID for where you see *yourAndrewId*.

START EARLY!

1 Check Out or Update Base Code and Libraries

Please make sure you have up-to-date libraries and course files before starting an assignment.

If you have not done working-directory set up as described in the first assignment (like in case you need to work from a different computer), please see Problem Set 1 and set up the working directory.

I assume you created the working directory called *24783* under your home directory and you checked out your Git repository in there.

Home directory is typically *C:\Users\username* in Windows, */Users/username* in macOS, and */home/username* in Linux, where *username* is the user name in your local computer.

First, open command-line (Developer PowerShell or Terminal), and move to your working directory by typing:

```
cd ~/24783
```

You need to check out (or clone) Git repositories once. If you have not checked out yet, do the following:

```
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/course_files.git
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

You need to replace "yourAndrewId" with your Andrew ID. You'll be asked to type in credentials.

Also we are going to use two additional repositories:

```
git clone https://github.com/captainys/MMLPlayer.git
git clone https://github.com/captainys/public.git
```

If you are successful, you should have the following directory structure under your home directory.

```
Your User Directory
├─ (Other files and directories)
└─ 24783
   └─ course_files
```

```
├── public
├── MMLPlayer
└── yourAndrewID
```

If you already have checked out these repositories (most likely you did for Problem Set 1), you need to update (or git pull) in those repositories. By change directory to the location where you checked out repositries and then type:

```
git pull
```

To update all four repositories, you can type the following commands in a sequence:

```
cd ~/24783/course_files
git pull
cd ~/24783/yourAndrewID
git pull
cd ~/24783/public
git pull
cd ~/24783/MMLPlayer
git pull
```

2 Set up SOLNP

CD to your 24783 directory and type the following commands to check out solnp and dlib.

```
      git clone https://github.com/KristerSJakobsson/solnp
cd solnp/library
git clone https://github.com/davisking/dlib
```

3 Copy Base Code and Add to Git's Control

Do the following to make a copy of the base code to your directory.

```
cd ~/24783
cp -r course_files/ps4 yourAndrewID/.
```

Adding /. after the destination directory is a weak defense to prevent files from copied to wrong location in case you misspelled the directory.

And then type the following to add them to the Git's control:

```
git add yourAndrewID/ps4
```

Once you add ps4 sub-directory to Git's control, you can do commit and push as many times as you want to send your files to the server with no penalty before the deadline.

It is recommended to make frequent commits so that you can go back to earlier version in case you mess up.

4 Make a CMake Project

4.1 Top-Level CMakeLists.txt

Write a top-level CMakeLists.txt under ps4 sub-directory, so that:

- it enables C++17 features,
- it includes ps4_1 sub-directory,
- it includes MMLPlayer library (optional), and
- it includes public libraries.
- it includes solnp library.

Since public-library sources are located outside of your source tree, you also need to specify where the build files are written. Therefore, the lines for including the MML player and public libraries should look like:

```
add_subdirectory(../../public/src ${CMAKE_BINARY_DIR}/public)
add_subdirectory(../../MMLPlayer/ym2612 ${CMAKE_BINARY_DIR}/ym2612)
add_subdirectory(../../MMLPlayer/mmlplayer ${CMAKE_BINARY_DIR}/mmlplayer)
```

When you use add_subdirectory, use relative path. Absolute path like C:/Users/soji/24783/soji does not exist in the grading environment. Also use slash instead of backslash. Windows can accept both backslash and slash. Other platforms only accepts slash. I want you to learn cross-platform programming.

To include solnp library, add the following:

```
add_subdirectory(../../solnp solnp)
include_directories(../../solnp/src)
```

4.2 CMakeLists.txt Files for Executables

Write a CMakeLists.txt file in ps4_1 sub-directory. It must define an executable called ps4_1, which uses ps4_1.cpp, data.h, data.cpp, ffd.h, and point.h. Don't forget MACOSX_BUNDLE keyword even if you are doing it in Windows or Linux to make it cross platform.

Target ps4_1 needs to link dlib and fssimplewindow libraries.

Also, there has been a case that include_directories specified in the top-level CMakeLists.txt is ignored in the subdirectory, which did not happen in my environment. But, if that happens to you, add the following line in the CMakeLists.txt of the sub-directory.

```
include_directories(${CMAKE_SOURCE_DIR}/../../solnp/src)
```

Once you set up CMakeLists.txt files, test-build ps4_1. First you create a build directory somewhere outside of ps4 directory, and then run cmake, and then type:

```
cmake --build . --target ps4_1 --config Release
```

to build the executable. If you omit –target ps4_1, you will need to wait for libraries unrelated to this assignment. If you run the ps4_1 at this time, you will see Fig. 1:

4.3 Add to Git's Control

After writing these files, make sure to add the files to Git's control.

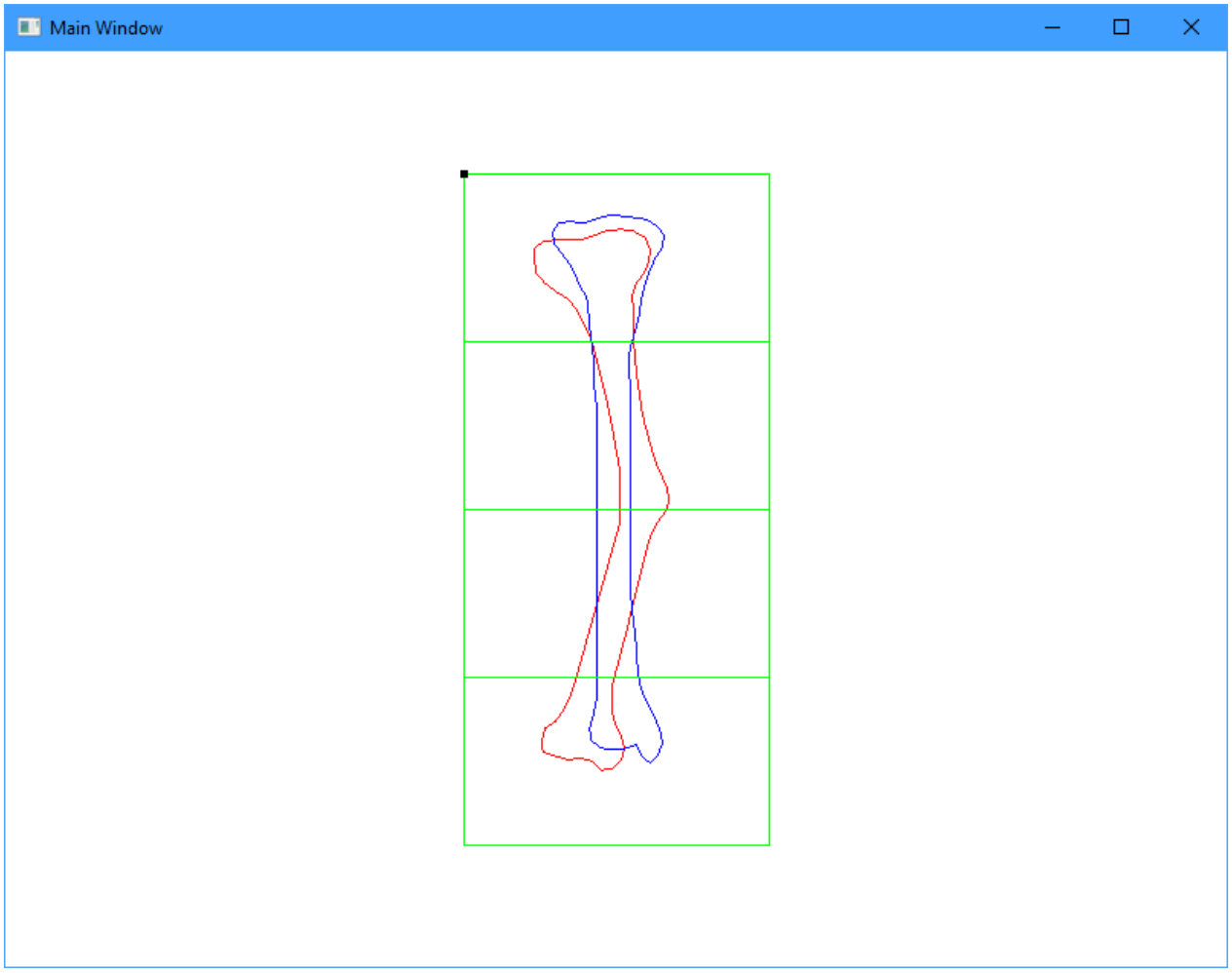

Fig. 1: Screenshot of the base code

5 Matching Bone Model with X-Ray

There is a medical condition in which the bone is severely deformed. It could happen when a patient had a broken bone but did not properly support by orthopedic cast while bone was healing. Once the bone is re-connected, it does not become straight by itself. Fig. 2 shows an X-Ray image of such a patient.

The doctor can correct a severely deformed bone by cutting it and applying an exoskeleton, and gradually adjust the position of the exoskeleton a few millimeters per day. However, to straighten the bone, the planning is very important.

A geometric model of the deformed bone would help the doctor plan the position adjustment. A CT scanner is commonly used for making a geometric model of a bone. However, the CT scanner has a cost and radiation problem.

Fortunately, it is not a random shape. A method for creating a geometric model of the deformed bone by matching a bone template to an X-ray image has been proposed. Taking one X-ray image is much cheaper and the radiation exposure is much less than CT.

It uses a method called free-form deformation. It first applies a structured grid over the template bone, and parameterize the XYZ coordinates of the template bone in the grid. Then, the optimizer deforms the grid, and then the coordinates of the template bone are moved to new locations by de-parameterization. Fig. 4 shows parameterization and de-parameterization example. (Free-Form Deformation code is given in the base code. You won't have to write your own Free-Form Deformation.)

For this optimization, the objective function is the difference between two polygons, one is the contour of the deformed bone in the X-ray image, and the other is the contour of the template bone deformed by the free-form deformation.

In this problem set, you need to:

1. write the objective function, which compares two polygons and returns the difference, and
2. make this program so that it shows how the optimizer deforms the contour closer to the final shape.

In the real setting, first the control points are optimized, and then the same free-form deformation is applied to the 3D model, and then do the same process from the different orientation. but we do not go that far in this problem set. We just find a good control points for free-form deformation from one viewpoint.

First read and understand the base code. You can play with Free-Form Deformation by N and P keys to move cursor forward and backward, arrow keys to move the control point.

The program needs to run the optimizer when the user presses the space key.

If you write a good objective function, the control polygons will look like Fig. 3. There can be a different solution.

The polygon of the template bone (which your optimizer needs to deform by the FFD) is tibia_template, and the contour of the deformed bone is tibia_patient defined in data.h. Both have NPoint points, but do not assume both are oriented equally, nor tibia_template[i] corresponds to tibia_patient[i].

The objective function is:

```
double compare(const int np1,const Point p1[],const int np2,const Point p2[])
```

in ps4_1.cpp. In the basecode, it always returns zero. The first polygon is defined by np1 and p1, and the second polygon is np2 and p2. The function is called from objective_function, which takes care of the deformation.

Fig. 2: X-Ray image of a patient

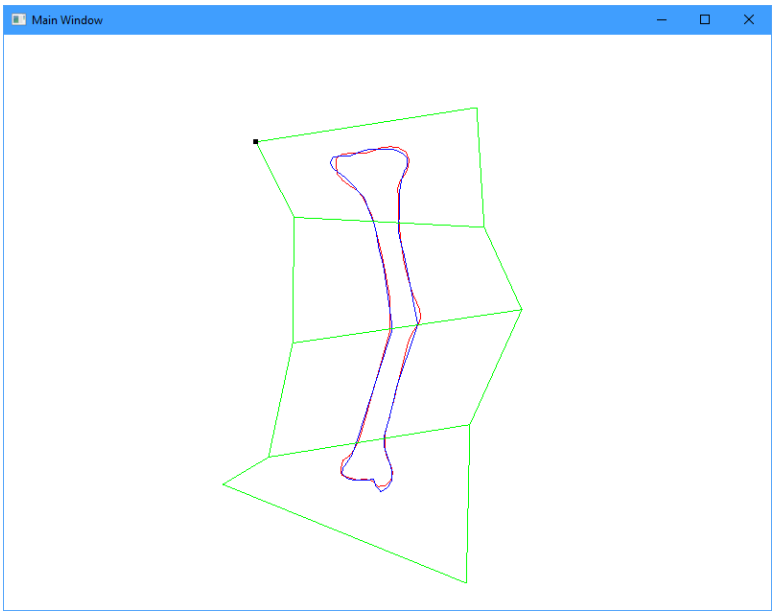

Fig. 3: Optimized Bone Contour

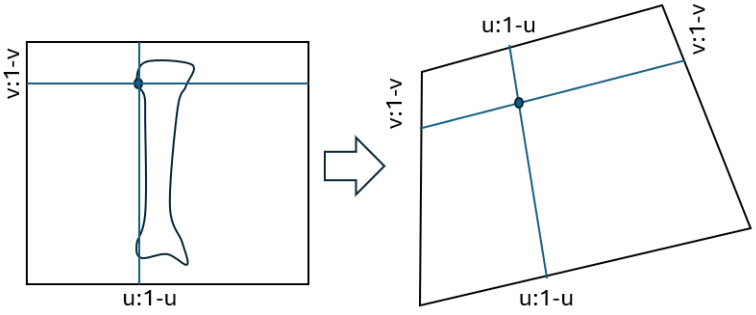

Fig. 4: Free-Form Deformation

Make sure to test your program with our compiler server. Your solution needs to pass the compiler server. When you test your solution, take a screenshot that SHOWS NO ERROR (no red lines) and save in your ps4 directory as sshot.png, and git-add the screenshots.

Good luck!

6 Test Your Code on the Compiler Server

Test your source files (.cpp and .h files) on the compiler server. Some assignment may not require .h files. You do not have to test files that you don't make modifications. The files you need to test are the ones you write or modify.

You need to take a screenshot of the compiler server showing "No Error" and submit to the git server. (See above.)

We have four compiler servers:

- http://freefood1.lan.local.cmu.edu
- http://freefood2.lan.local.cmu.edu
- http://freefood3.lan.local.cmu.edu
- http://freefood4.lan.local.cmu.edu

Compiler servers are accessible from within CMU network only. To access from the outside network, use VPN to connect to the CMU network.

Make sure you don't see red lines when you select your files and hit "Compile" button on the server.

We have multiple servers to make it less likely that all of them need to shut down for maintenance. If do not have to test on all of the servers. You need to make sure that your code passes on one of the servers.

In this assignment, you need to select ps4_1.cpp and data.h and click on the Compile button as shown in Fig. 5.

7 Submit

Lastly, you need to submit using git. What you need to do are two things: (1) add files to Git's control, and then (2) send to the git server.

7.1 Add Files to git's control

In this case, you want to add all the files under ps4 subdirectory. To do so, type:

```
git add ~/24783/yourAndrewID/ps4
```

This command will add ps4 directory and all files under the subdirectories.

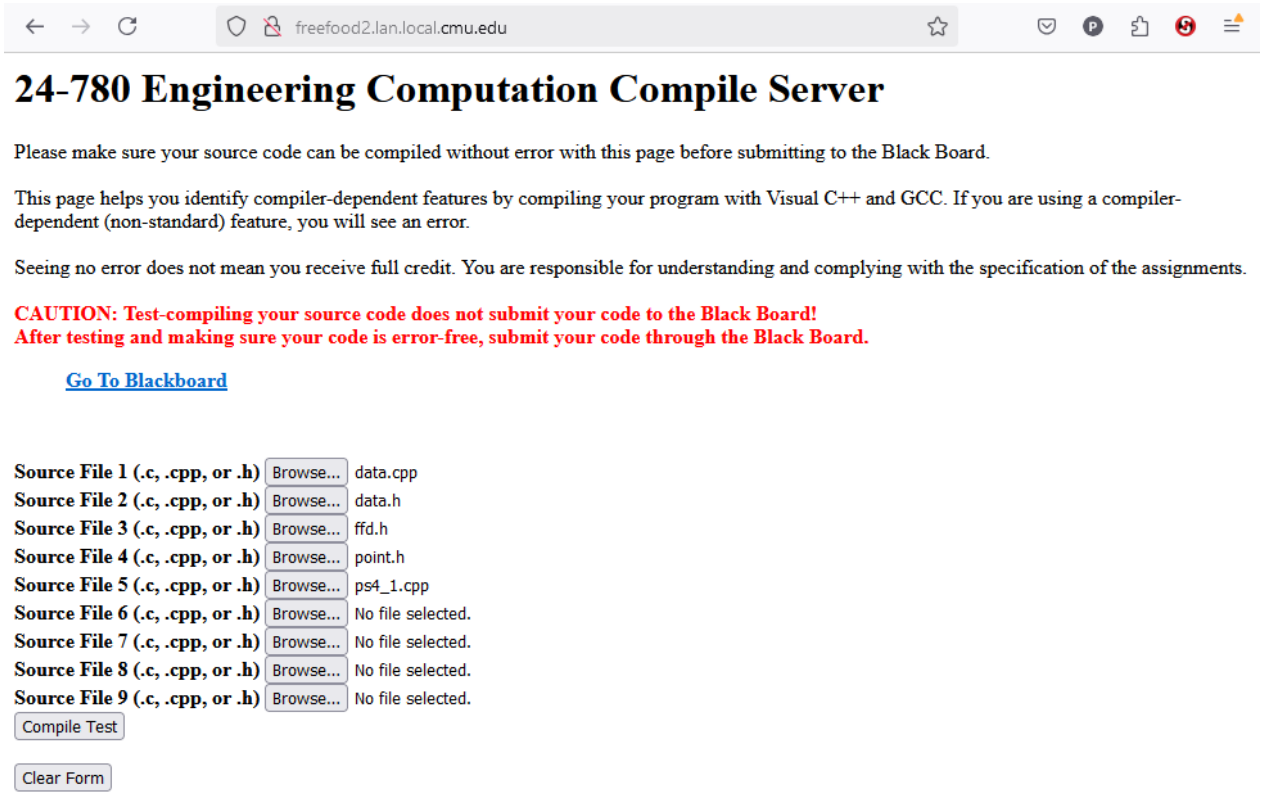

Fig. 5: Freefood Server

7.2 Send to the Git Server

In Git, sending files to the server is a two-step process. The first step is local commit. You can do it by:

```
git commit -m "Problem Set X solution"
```

The message can be anything, but it is recommended to type something meaningful, at least you can see what changes you made to your repository.

Local commit is just local. Git server does not know about any local commit unless the commit is sent (or pushed) to the server. To do so, type:

```
git push
```

Make sure to do it in the CMU network. If you are working from home (probably most likely), use VPN to connect to the CMU network.

You can re-submit (commit and push) your solution as many times as you want with no penalty before the submission due.

8 Verification

Clone your repository to a different location and make sure that all of your files have been sent to the Git server.

You can do the following:

```
cd ~
mkdir 24783Verify
cd 24783Verify
git clone https://yourAndrewID@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

Once you made sure all the files have been submitted, you can delete files and directories under 24783Verify directory.