MPHG002: Research Computing With C++
Assignment 1: Point and Surface Based Registration

Assignment Presentation
For this coursework assignment, you are expected to submit a short report and your code. The purpose of the report is to answer the questions below, to present your results and provide a brief description of your design choices and implementation. The report need not be very long or overly detailed, but should provide a succinct record of your coursework. The report must have a cover sheet stating your name, your student number, and the code of the module (MPHYG002). Marks will be deducted for poorly commented and difficult to understand code at the discretion of the marker. It is also worth bearing in mind that most of the marks for each task are awarded for a sensible attempt even if the code does not work correctly or the final numerical outputs are incorrect.

Submission
You should submit your report and all of your source code so that an independent person can run the code. All coursework should be submitted electronically by email directly to m.clarkson@ucl.ac.uk. The source code can be zipped up and emailed along with your report. There is no need to include your source code in your report, but you can refer to it and if necessary reproduce lines if it helps to explain your solution. The deadline for submissions is 5pm on Tuesday 17th March. An acknowledgement of receipt will be sent out after each submission.

Preliminaries
The aim of this coursework is to implement two working pieces of software. The two examples have been chosen to be somewhat numerical, and able to be implemented with the taught libraries from lecture 3. At this stage of the course, single-threaded execution is all that is required.

You are provided with skeleton code at this Github location:
https://github.com/UCL-RITS/research-computing-with-cpp-demo.git

The purpose of this skeleton code is mainly to provide a CMake framework for downloading versions of Eigen, Boost and ITK (although this coursework does not use ITK), compiling them, and making them available for the rest of the questions. Any errors discovered in the provided CMake code and examples will be pushed to the master branch of this repo, and will be notified to students via Moodle.

Reporting Errors
You can post questions to the Moodle Discussion, if you believe that there is a problem with the example CMake code, or the build process in general. To make error reporting useful, you should include in your description of the error: error messages, operating system version, compiler version, and an exact sequence of steps to reproduce the error.

Point Based Registration

1. Read “Least-Squares Fitting of Two 3-D Point Sets” Arun et. al. PAMI, 1987, 10.1109/TPAMI.1987.4767965. Also available to download from Moodle.

[0 marks]

2. Take a fork of https://github.com/UCL-RITS/research-computing-with-cpp-demo.git and create a new class/function to implement a Point Based Registration according to Arun’s paper, using Eigen. The method should output to console the Fiducial Registration Error (FRE), which is the Root Mean Squared (RMS) distance between corresponding points once aligned. The final output of the registration should be a 4x4 matrix. Justify your choice of design of class/function, the input and output parameters.

[20 marks]

3. Create unit tests for the Point Based Registration class/function using the Catch framework taught in lecture 1. Consider the basic successful registration cases, and cases with no data, not enough data, or unequal number of points. Errors should be raised using your own custom exception type, so implement your own exception class. Make sure your registration class/function documents the use-cases, and/or throws meaningful errors as appropriate. Generate your own test data.

[20 marks]

4. Create a new program entry point for a “Point Based Registration” program, with a name of your chosing, with a `main()` function.

[2 marks]

5. Use Boost.Program_options to provide command line options. The program should provide the options to read two files containing point data and write the resulting matrix to a third file. If you can’t use Boost, you may write this part yourself using standard C++, but be aware that marks will be deducted if the command line parsing is not robust.

[8 marks]

6. Write code to read a list of points from a file. The file format should be plain text, reading successive x, y, z coordinates for each point. Read: http://www.tomdalling.com/blog/software-design/resource-acquisition-is-initialisation-raii-explained/ Use RAII to make sure that file handles are always closed. Write unit tests to test your file access.

[10 marks]

7. Connect your components together, so that your `main()` program can parse the command line arguments, read the points from file, do the registration and output the matrix to file. Test Data can be found in folder `research-computing-with-cpp-demo/Testing/PointBasedRegistrationData`. Create a unit test that reads files `fixed.txt` and `moving.txt` as input, calculate a 4x4 transformation matrix, and compares the output matrix to `matrix.4x4` to 3 decimal places. The unit test must pass for full marks. Paste your final matrix into the report.

[5 marks]

8. In medical imaging, point-based registration would normally have a relatively small number of points. Discuss how your program would currently work with point sets over 100,000 points. How efficient would it be? What might you change to make your solution scale up? You are not required to make these changes.

[5 marks]

Surface Based Registration

9. Read “Method for Registration of 3D Shapes”, Besl and McKay, PAMI 1992. Also available to download via Moodle.

[0 marks]

10. Surface based registration can be achieved by registering two point sets. A very simple implementation, written in pseudo-code might read:

```
initialise a transformation (normally a 4x4 matrix containing rotations and translations)
do {
        for each point in one point set,
                find the closest corresponding point in the other point set after it has
                been transformed by the current transformation

        use point based registration (Question. 2) to get a
        transformation aligning the points.

        update the current transformation
} while RMS is still decreasing
```

Implement such a Surface Based Registration algorithm in a class/function. Justify your choice of design, including your choice of input and output, and also the efficiency of the implementation.

[20 marks]

11. Add error handling, to your surface based registration by throwing exceptions, using the same exception class as you created in Question 3.

[5 marks]

12. Connect your components together. Create a program entry point for “Surface Based Registration”, with a name of your chosing. Design and implement command line parsing. The program should parse the command line, load two surfaces (represented as large point sets), perform basic error checking, and use the surface based registration method of Question 10, to calculate an output matrix representing the alignment of the point sets. Test data can be found in `research-computing-with-cpp-demo/Testing/SurfaceBasedRegistrationData`. Create a unit test to read `fixed.txt` and `moving.txt` and register the points in `moving.txt` to the points in `fixed.txt` and output the transformation matrix. The matrix should be similar to the provided `matrix.txt`, and depends on your exit criteria in the loop. The unit test should simply measure the final RMS and test that it is below some threshold. The two point sets `fixed.txt` and `moving.txt` are generated by transforming `fixed.txt` into `moving.txt`, so there is no additional noise added, so the threshold should be low. What is the benefit of providing this type of unit test, and what is this type of testing also called.

[5 marks]