A1: Calibration

1. Overview

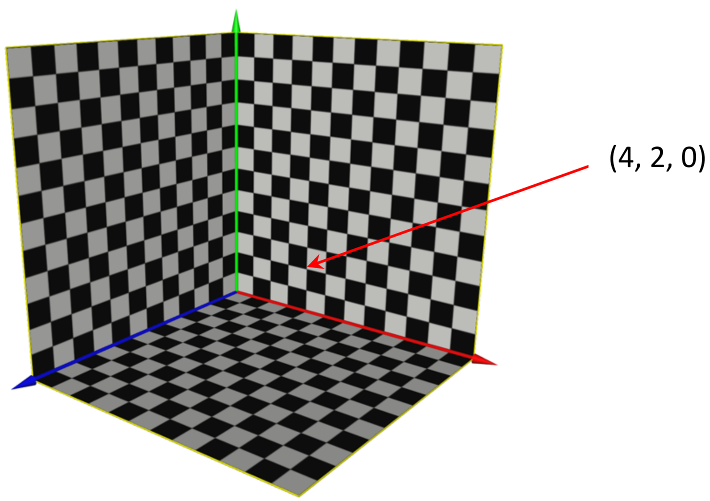
In this assignment, you will gain hands-on experience in camera calibration. Specifically, you will practice:

1) Basic linear algebra operations;
2) Setting up and using a (virtual) apparatus to collect a set of 3D-2D corresponding points;
3) Computing the SVD and inverse of a matrix;
4) Extracting intrinsic camera parameters;
5) Extracting extrinsic camera parameters;
6) Using open-source libraries to visualize and validate your calibration results.

2. The tasks

Task #1: Collect at least 6 pairs of 3D-2D corresponding points (10%)

Compile and run the viewer. You will see a virtual calibration rig as follows



In this rig, the **red**, **green**, and **blue** arrows denote the ***X***, ***Y***, and ***Z*** axes, respectively. With the grid texture, you can easily figure out the 3D coordinate of any points on the grid.

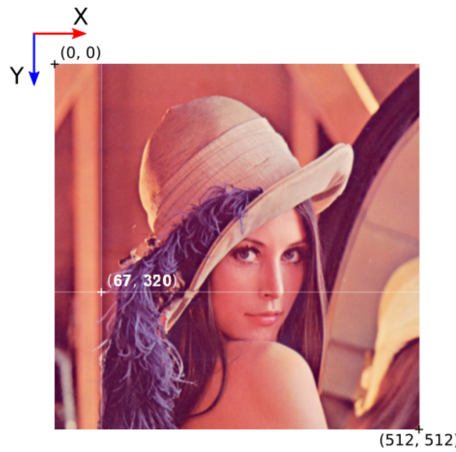
You can use your mouse to manipulate the scene (i.e., move and reorient your camera).

Take a picture of the scene by pressing the "s" key. Then you will see a dialog asking for a file name. You can provide any name, e.g., "image_01.png", and save it to your disk.

With the image, pick ***n (n>=6)*** points on the virtual scene, and figure out their corresponding image points. To make the access to the data easier, please store these correspondences in a "***.txt***" file in which each line has 5 coordinates separated by space, i.e., the first 3 coordinates represent a 3D point and the subsequent 2 coordinates represent the corresponding 2D image point. See the two examples in "/resources/data".

Data: use the provided data for testing your camera calibration algorithm. In your submission, you must include the test results using your collected 3D-2D correspondences. We assume every team collects different data for the experiment.

Image coordinates: The image below shows the image coordinate system. Image coordinates are denoted in pixels, with the origin point (0, 0) corresponding to the top-left corner of the image. The X-axis starts at the left edge of an image and goes towards the right edge. The Y-axis starts at the top of the image towards the image's bottom. All image pixels have nonnegative coordinates.



Directions and pixel positions in the image coordinate system.

Task #2: **Implement the camera calibration algorithm.**

In the file "/A1_Calibration_Code/Calibration/calibration_method.cpp", the function "calibration" is defined but not implemented. Comments and guidance are left in the function. Example code is also provided within the function for you to get familiar with the necessary data structures and algorithms.

Camera calibration can be achieved by tackling the following subtasks:

- Construct the P matrix (10%).
- Solve for M (the whole projection matrix, i.e., $M = K * [R, t]$) (20%).
- Extract intrinsic parameters from M (10%).
- Extract extrinsic parameters from M (10%).

Task #3: **Evaluation**

- Come up with a method to evaluate the quality of a calibration result (5%).
- Invite student(s) of another group to review your code. Incorporate the received feedback in your implementation and reflect on the changes in the report. (5%).

Submission

Your submission should include:

- **A report** (30%). The report should be as concise as possible (preferable ≤ 3 pages excluding figures, tables, and references), but it should provide sufficient information to reimplement your method to reproduce your results, and it should include:

- Test results of your algorithm on both the provided data and your data.
- Snapshots of your results put next to the input images.
- Explanation on how you verify the intermediate result of each step.
- Discussion on how you determine the sign of ρ.
- Discussion of the accuracy of the result and how to improve it.
- Reflection on how the feedback received from the other group helped or improved your implementation.
- A short description of "who did what".

- Data

- Your input 3D-2D point pairs (in *txt* format).
- A screenshot of the input image.

- Source code

- The entire source code including the provided code framework and your implementation of the '*calibration (...)*' method.
  - Your source code should compile and reproduce your results without modification.
  - Please do NOT include the 'build' directory (which contains the intermediate files in a build step).

Please compress all the above into an archive file and name it in the following format:

GEO1016_Assignment_1_Group_X.zip

where 'X' is your group ID, which can be found here:

https://docs.google.com/document/d/1WMPXgWD0_2F9oDSub1K-g6NdRKqIRyWj3sUFDCpfFSk/edit

Appendix: About the code framework

The code framework is based on *Easy3D*[1]. The viewer provides visualization of 3D scenes and cameras. It also serves as a means to intuitively validate the camera calibration results.

The usage of the viewer:

Left button:	*rotate the camera*
Right button:	*move the camera*
Key 's':	*snapshot (i.e., take a picture)*

[1] https://github.com/LiangliangNan/Easy3D. The code framework uses a stripped earlier version of Easy3D, which is not compatible with its latest version.

Key 't': *show/hide the virtual camera*

Key 'space': *run the camera calibration algorithm*

I assume you have some experience with C++ and you can easily compile and run the code of this assignment. If not, please have a look at:

https://3d.bk.tudelft.nl/courses/geo1016/resources/build_C++.html

The assignments are not designed for testing you, but to help you gain knowledge. You will get every support to accomplish all assignments. Forget the mark and enjoy the process!!!