A2: Triangulation

1. Overview

In this assignment, you will gain hands-on experience in reconstructing 3D geometry from corresponding image points of two images. Specifically, you will practice:

- Basic linear algebra operations;
- Estimating fundamental matrix F from corresponding image points;
- Recovering relative pose (i.e., R and **t**) from the fundamental/essential matrix;
- Determine 3D coordinates of image points using the recovered relative pose.

2. Methodology

The underlying theory has been explained in the lecture and is elaborated in the lecture note (i.e., handout). In this section, mainly the major steps of each algorithm are provided. Please refer to the lecture note and/or the lecture slides for the technical details.

Step #1: Estimate fundamental matrix F from given corresponding image points using the normalized 8-point algorithm **(20%)**

- Normalization. – 5%
- Linear solution (based on SVD). – 5%
    - The recovered F is up to scale, please make the last element 1.0
- Constraint enforcement (based on SVD). – 5%
    - Find the closest rank-2 matrix
- Denormalization. – 5%

Step #2: Recover relative pose (i.e., R and **t**) from the fundamental matrix **(20%)**

- Find the 4 candidate relative poses (based on SVD). – 10%
- Determine the correct relative pose. – 10%
    - determinant(R) = 1.0 (within a tiny threshold due to floating-point precision)
    - most (in theory it is 'all' but not in practice due to noise) estimated 3D points are in front of the both cameras (i.e., z values w.r.t. camera is positive)

Step #3: Determine the 3D coordinates for all corresponding image points **(30%)**

- Compute the projection matrix from K, R, and t. – 10%
- Compute the 3D point using the linear method (based on SVD). – 10%
- [Optional] Non-linear least-squares refinement of the 3D point computed from the linear method. – **10% extra**
- Triangulate all corresponding image points. – 10%

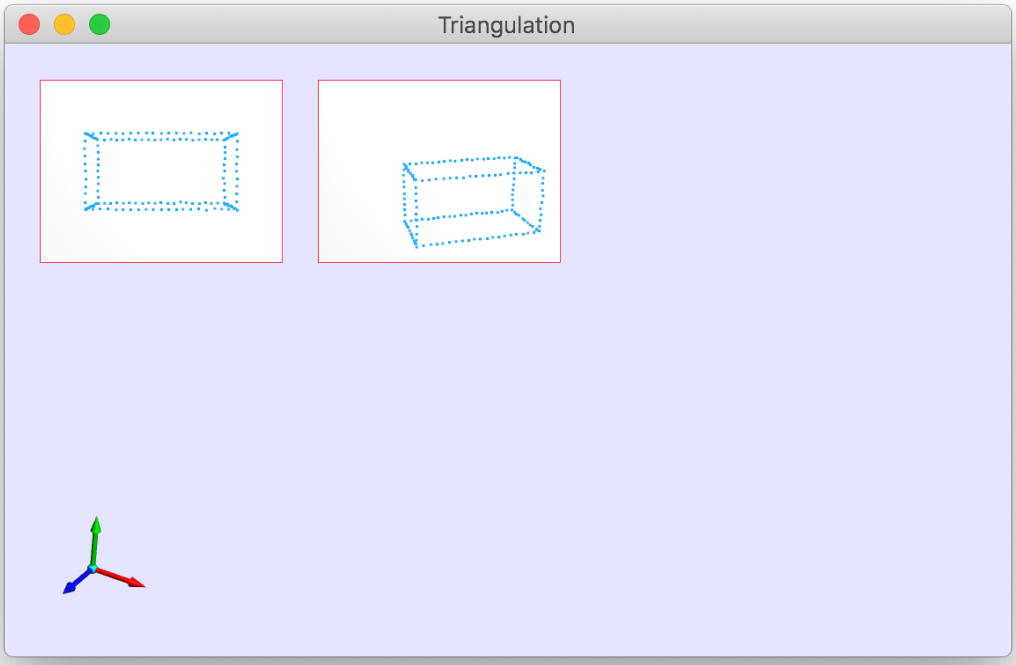
It is strongly advised to **define a function for each sub-task**. This way you have a clean and well-structured implementation, which also makes your testing and debugging easier. Each function may also be reused for other purposes.

4. Data, code framework, and viewer

The data used in this assignment are 160 pairs of image points from two images. The points from the two images have one-to-one correspondences. All points are stored in homogenous coordinates (i.e., in the form of *x, y, w* and *w* is set to 1.0). Each line has three floating-point numbers representing a single point. The points of the two images are store separately in:

- /A2_Triangulation_Code/resources/data/image_points_0.xyz
- /A2_Triangulation_Code/resources/data/image_points_1.xyz

Build and run the viewer from the provided source code, the image points will be automatically loaded for you (so you don't need to do anything for file IO). You will see the two images (in white containing blue image points) visualized as below:



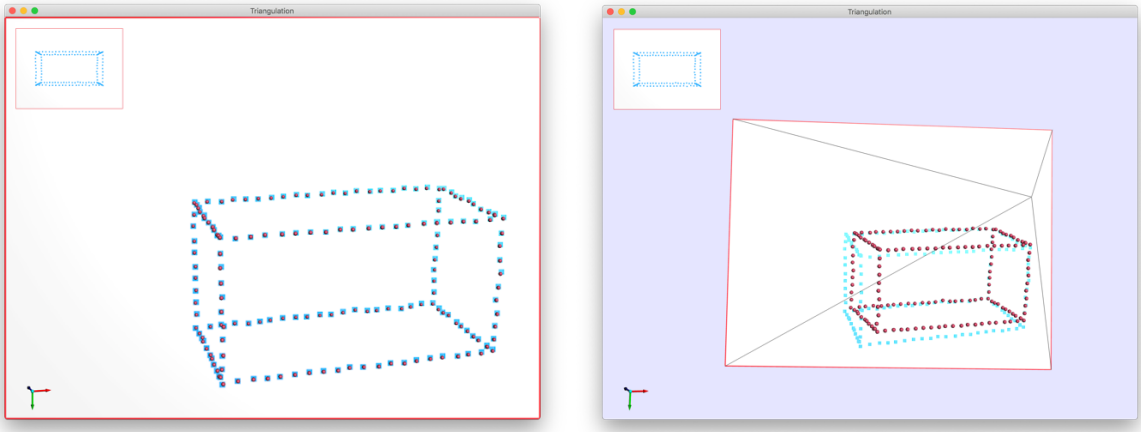
If you **press the 'space' key, the reconstruction method will be triggered**.

The reconstruction function is named '*triangulation(...)*', located in the file '*triangulation_method.cpp*'. You will need to implement this function to reconstruct 3D points from the corresponding image points. Comments and guidance are provided in this function. Example code is also provided within this function for you to get familiar with the necessary data structures and APIs.

In addition to the interactions (e.g., rotate, translate, zoom in/out) provided by the viewer, you can also

- take a snapshot of the visualization by pressing the ‘s’ key.
  (It is required to include in your report a few snapshots of the reconstructed 3D points from different views).
- save the reconstructed 3D points into a file by pressing both ‘Ctrl’ and 's'.
  (It is required to include the file of your reconstructed 3D points in your submission).

When the ‘*triangulation(...)*’ function has been correctly implemented (please refer to the comments in the code), running the method will also automatically update the viewer so the reconstructed points will be visualized using the recovered camera extrinsic parameters. Such visualization can help you validate your result.



(a) 3D points visualized using the recovered R and **t**. (b) 3D points visualized in another view

3. Submission (deadline: 23:59, Mar 19)

Your submission should include

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

(2) A report **(30%)**
   - Description of the methodology. – 5%
   - Demonstration of your result. – 5%
   - Evaluation of your results (e.g., accuracy, potential ideas to improve it). – 5%
   - Discussion on how to obtain the camera intrinsic parameters (fx, fy, cx, and cy; or simply K) and the effect of errors in the intrinsic parameters on the final reconstruction. You can tune the parameters of K (in the function

Triangulation::key_press_event(…) in '*triangulation.cpp*') and run the reconstruction multiple times to evaluate and compare the results. – 10%

- A short description of "who did what". – 5%

(3) A file of the reconstructed 3D points (in .xyz format. You can press 'Ctrl' and 's' to save).

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

GEO1016_Assignment_2_Group_X.zip

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

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

[Liangliang]: The assignments are not designed for testing you, but to help you gain knowledge. You will get every support to acommplish all assignments.

So forget the mark, ask questions, and enjoy the process!!!