A2: Triangulation

1. Overview

In this assignment, you will gain hands-on experience in reconstructing 3D geometry from the 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 $\mathbf{t}$) 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 notes (i.e., handout). In this section, mainly the major steps of each algorithm are provided. Please refer to the lecture notes and/or the lecture slides for the technical details.

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

- Normalization. (5%)
- Linear solution (based on SVD). (5%)
- Constraint enforcement (based on SVD). (5%)
    - Find the closest rank-2 matrix.
- Denormalization. (5%)

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

- Find the 4 candidate relative poses (based on SVD). (10%)
- Determine the correct relative pose. (10%)
    - The correct (R, $\mathbf{t}$) pair is the one that yields the most (in theory it is 'all' but may not in practice due to noise) 3D points lying in front of both cameras (i.e., z values w.r.t. both cameras are positive). See **Step #3** for 3D geometry recovery.

Step #3: Determine the 3D coordinates for all corresponding image points **(20%)**

- Compute the projection matrix from K, R, and $\mathbf{t}$. (5%)
- Triangulate all corresponding image point pairs using the linear method. (15%)
- [Optional] Non-linear least-squares refinement of the 3D points computed from the linear method. You must refine the coordinates of all points simultaneously (instead of one by one by calling the same function multiple times). **(10% extra)**

Step #4: **Evaluation (10%)**

- Come up with a method to evaluate the reconstructed 3D points (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%).

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.

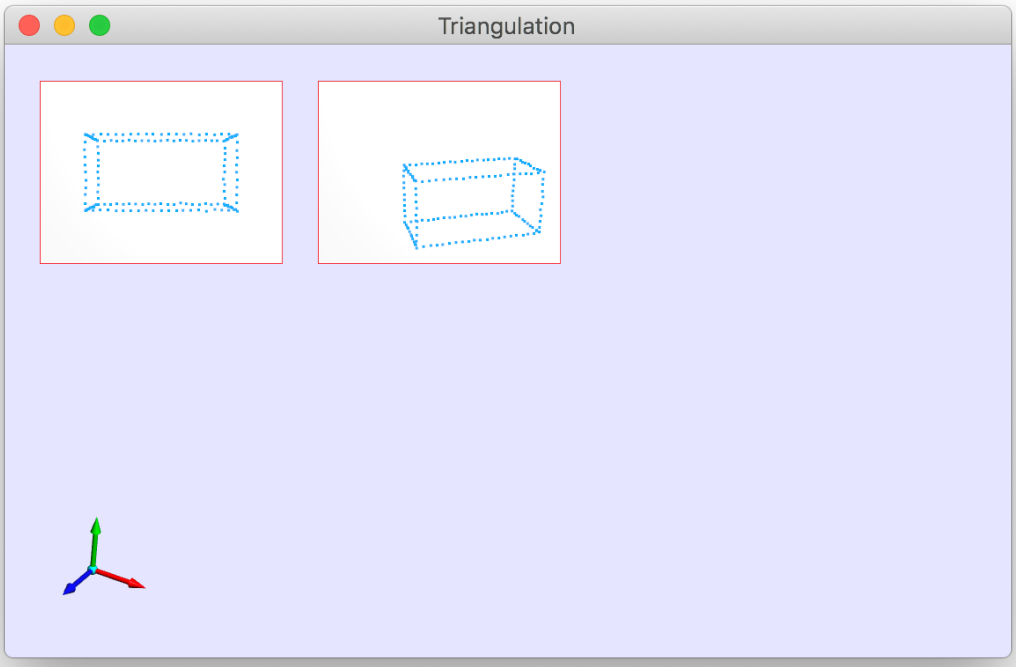
Note: You're encouraged to discuss and share experiences with other groups (but coping codes from others will be penalized) and discuss what you have learned in the report.

3. 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 stored 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 as follows (i.e., two white images containing blue points):



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

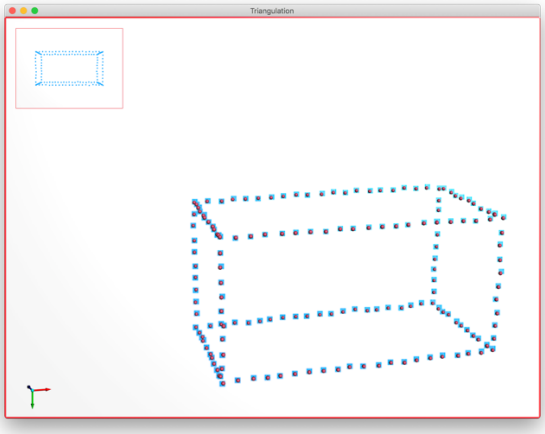
The reconstruction function is named '*triangulation(...)*', located in the file '*A2_Triangulation_Code/Triangulation/triangulation_method.cpp*'. You will need to implement this function to reconstruct 3D points from the corresponding image points. Comments and guidance are provided in this function. 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

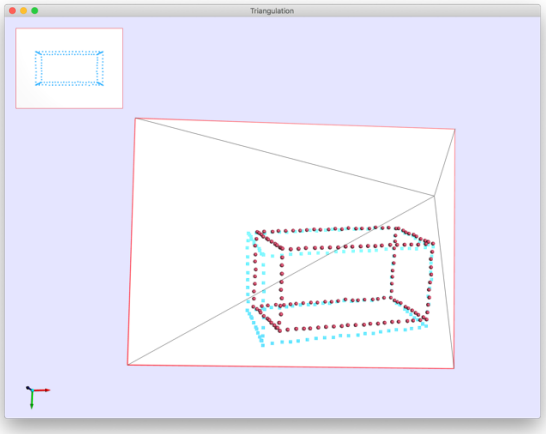
- take a snapshot of the visualization by pressing the 's' key.
  (It is required to include in your report a few snapshots of the reconstructed 3D points from different views).
- save the reconstructed 3D points into a file by pressing both 'Ctrl' and 's'.
  (It is required to include the file of your 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 $\mathbf{t}$.



(b) 3D points visualized in another view

4. **Submission**. Your submission should include:

(1) **Report** (max 3 pages excluding figures and tables) **(30%)**

You're expected to deliver a serious scientific report. Make every sentence, equation, and notation precise and clear to avoid misinterpretation. Meanwhile, the report should be as concise as possible, but it should provide key information to reimplement the method to reproduce your results, and it should include:

- Description of the methodology. We encourage the use of mathematical languages (e.g., equations). (5%)
- Demonstration of your result. (5%)
- Discussion on the quality of the reconstruction. Is there a way to quantitatively measure its accuracy? (5%)
- Can you still improve the reconstruction result (assuming the camera intrinsics and image matching are both accurate)? (5%)
- In this assignment, the ground-truth intrinsic parameters are given. In practice, how can you obtain accurate camera intrinsic parameters? Introduce a certain level of inaccuracy (e.g., 5%, 10%, 20%) in these parameters (in the following function: '*Triangulation::key_press_event(...)*' in '*triangulation.cpp*') and describe how it affects the reconstruction result. (5%)
- Reflection on how the feedback received from the other group helped or improved your implementation. (5%)
- A short description of "who did what". Please be specific.

To promote formal scientific writing, the following rules apply for assessment:

- Any misunderstanding or misconception of main concepts: 10% deduction.
- Multiple unclear or ambiguous descriptions: 10% deduction.
- Multiple typos, grammar issues, format issues (e.g., a figure/table without a caption or multiple figures/tables with the same caption, unindexed or unreferenced figures/tables), consistency issues (e.g., upper case and lower case used interchangeably, normal font and italic font used interchangeably), misuse of symbols in notations: 10% deduction.
- The report is more than half a page over length: 10% deduction.

(2) Data

- A file containing the reconstructed 3D points (in the *xyz* format). You can press 'Ctrl' and 's' to save your reconstruction result after triangulation.

(3) Source code

Please submit only the following file:

- *A2_Triangulation_Code/Triangulation/triangulation_method.cpp*

This file contains your implementation of the '*triangulation(…)*' function.

Your source code should compile and reproduce your results without modification. It is also recommended to test your code on different operating systems, as your code may be evaluated on an OS different from that it was developed.

Please compress the above files into an archive file named in the following format:

GEO1016_Assignment_2_Group_XX.zip

where 'XX' is a 2-digit number of your group ID. **In case you have an updated version, please append "_V2" to the title of your submission.**