CS635: Assignment #2 – Geometric (and Photometric) Stereo Reconstruction

Out: February 7, 2025
Due: February 21, 2025

Objective

The objective of this assignment is for you to reconstruct a significant fragment of a 3D object of your choosing using geometric stereo. This assignment will build off the previous assignment and using your calibrated camera, you will take 3 or more pictures of an object, reconstruct its geometry, and triangulate the computed surface. Remember: precision is very important -- small errors can lead to big calibration errors.

This assignment includes a significant “extra credit” which is to perform a simple photometric stereo reconstruction.

This assignment requires some mental planning work, some lab work, and some programming. The whole assignment is not that much work and mostly requires you to understand the process. I do recommend STARTING EARLY! (i.e. -- you will not be able to complete the assignment if you start the day before its due --).

Detailed Description

Step 0 – Image Capture and External Pose Estimation

Using your internally calibrated camera, **take at least 3 pictures** spanning an angular viewing area in front of your object. Since your camera is now internally calibrated, you only need to determine the camera’s external pose. You should choose a few well-chosen and known 3D points across all images and use them to perform pose estimation (e.g., if you put the checkerboard in the background and an object in the foreground, you can use points on the checkerboard to do pose estimation). You may implement a closed-form (linear) pose estimator or you can use an iterative optimization. The optimization can be of a general nonlinear form or a linear form (with some assumptions of locality). Also, if you take your pictures using a smoothly changing viewing position, you can use the pose of the previous picture to initialize the pose of the new picture. To do this, review the camera calibration lectures and assignment.

You should also plan on taking your pictures wisely – they should span a wide field-of-view of the object you choose. The object you choose should also be adequate for the reconstruction, both in terms of its colors (for point selection) and in terms of its aspect-graph (for preventing too many disocclusions). Also, it is probably easiest if all your to-be-selected-scene-points are visible in all images, however this is not necessary. For example, you can take pictures from one semi-circular path in front of the object.

Step 1 – Reconstruction (50%)

To perform the reconstruction of the object, correspond selected scene points on the object over at least 2 images (use the mouse) and reconstruct them. You might have points corresponded over 2 images up to all 3 or more images. Regardless, you can calculate their 3D position using a single formulation for linear reconstruction as explained in class. The more points you correspond, the more 3D data you have. You should have **at least 30 points** reconstructed between any pair of captured images. However, this number is rather arbitrary -- it should be “as many as you can get”. Thus use it as a minimum guideline, not as a goal. In general, the goal is for the front half of the object you choose to be “well reconstructed”.

Step 2 – Triangulation (25%)

To demonstrate the object, render a triangulated 3D reconstruction of the object. The virtual viewpoint will typically be somewhere in front of the object and the image from closest captured viewpoint to the virtual viewpoint should be used to triangulate the reconstructed points.

You may use a virtual trackball style interface to enable changing the virtual viewpoint when rendering the reconstruction.

Step 3 – Visualization (25%)

Your program should include basic GUI tools so that the source images, the clicked on features, and the triangulated 3D reconstruction can be selected to be viewed in a reasonable way.
NOTE: At least one of the captured images should be texture-mapped onto the triangulation.

Step 4 – Extra Credit (+50%)

For the same object, you can also perform a photometric stereo reconstruction with either the known lights assumption or the unknown lights assumption. There are various methods and you can use any of them, and also Matlab to solve/simplify the relevant equations.

Place your camera seeing your same object head-on. Then, turn off all light sources in the room and move a portable light to three (or more) positions surrounding your camera but looking at your object (e.g., at 10 o’clock, 2 o’clock, and 6 o’clock). You should take at least three pictures but taking more might be better (e.g., take 5 pictures and ignore the brightest and darkest value in each pixel).

If you use the known lights formulation, you can estimate the relative light vectors.

In all cases, you should place the lights at about the same distance from the object so that the maximum light intensity incident on the object is about the same.

Then, you can compute the per pixel normals. Once they are computed, the demonstration is to then support a ‘virtual light direction’ with your GUI and you can re-light the image of your object effectively simulating a wide range of light directions. You do NOT need to reconstruct the geometry of the object.

Note, photometric stereo can be in color but for this extra credit, you can simplify all images to grayscale if you like. To support color, you can i) convert color to grayscale, compute albedo, then return to color, or ii) compute albedo per channel. Note that there are more sophisticated methods.

Tools

To help with this assignment, I will provide you with a *2D triangulation library* ("triangulation") and with a *linear least squares optimization* ("LSQR") package. The former can be used to triangulate the reconstructed scene points using their projected positions on an captured image plane. It came from http://ect.bell-labs.com/who/sjf/. The latter can be used to compute a linear 3D reconstruction of scene points using known camera matrices or for help with photometric stereo. The latter package is intended for use in "large sparse linear systems" but you can equally use it for smaller dense linear systems. It is from http://web.stanford.edu/group/SOL/software/lsqr/. I have included "lsqr-simple.zip" which I have used before and a newer version called "lsqr++.zip".

Grading/Demonstration

Your demonstration will consist of you showing me your program in short demo session to be arranged (in my office). On or before the due date, please give Brightspace a zip file with a single directory called **"<your-name>-asgn2**" containing:

- Executable
- Data files (i.e., images)
- Other necessary files, DLLs, etc…

During your demo session, I will use the provided zip-file to grade your program. Your grade will be influenced by how well your particular camera/object is reconstructed, by the presentation and usability of your program, and by how well you complete the assignment requirements.

In this assignment, you may collaborate only to help with the mechanics of the assignment (e.g., using the pad, your camera, etc). *Everybody must take their own pictures*! Practically speaking, this means that nobody should have the same pictures or calibration results.

If you have questions, please come see me ASAP – do not wait until the last moment. Have fun and good luck!