ELEC 5630 Project 1

Photometric Stereo

Professor: TAN, Ping

Due Oct. 12, 2025

1 Photometric stereo

1.1 Task

For each object, we provide 16-bit integer PNG images with the resolution of 612×512 from 96 different lighting directions. All the given images are linearized, i.e., the pixel value is the radiance predicted by the reflectance model. The mask image ('mask.png'), lighting directions ('light_directions.txt', 3×96, with each row as a unit 3D vector), lighting intensities ('light_intensities.txt', 3×96, with each row representing the intensities in RGB channels; images are required to be normalized by dividing these intensity values per channel before performing photometric stereo), and image file names ('filenames.txt') are provided within the subfolder of each object.

We provide a codebase in Python as a warnstart. We have included a main script named mainBaseline.py that reads images from a directory, calls the photometric stereo function (that you will be implementing), and generates images showing the output and some of the intermediate steps. You are free to modify the script as you want, but note that your script mainBaseline.py should be executable for our evaluation. Please ensure your code runs correctly with the original script and generates the required output images.

Every script and function you write in this section should be included in the *python/* directory. Please include the resulting images in your write-up.

1.2 Method

1.2.1 Least Squares-Based Method (7 pts)

Estimation of normal: Implement the least squares-based Photometric Stereo using the traditional Lambertian reflection model for the data we provided. Test the algorithm and obtain normal maps. You can refer to the code on GitHub, but self-implementation is encouraged.

Dealing with shadows and highlights: Shadows and highlights break the linear Lambert's model. A simple solution to this problem is to sort all the observations at each pixel and discard a certain percentage of the darkest and brightest pixels to eliminate shadow and highlight, respectively. After discarding those noisy observations, the Lambertian photometric stereo algorithm can be applied to the remaining data. Implement this method and test

it with the provided data.

1.2.2 Low-rank Factorization (PCA) (6 pts)

Implementing the low-rank matrix factorization algorithm to solve Photometric Stereo using the traditional Lambertian reflection model for our provided data. Test the algorithm and obtain normal maps. You can refer to the code on GitHub, but self-implementation is encouraged.

1.2.3 From normal to mesh (7 pts)

Based on the content in the slides, the surface height at any point can be recovered by integrating along the path. Implement the Frankot-Chellappa algorithm that recovers depth from the normals. After obtaining the depth map, you can use 'create_from_point_cloud_poisson' in 'open3d.geometry.TriangleMesh' or an alternative function in the Python package to convert the depth map into a mesh. We will check output mesh qualities based on MeshLab.

2 Submission

You only need to upload **ONE** zip file containing the code, the README file, and the results. The code should be in Python format. Implementing another language will be skipped, and get 0 pts in this project.

Many of the algorithms you will be implementing as part of this project are functions in the Matlab image processing toolbox. You are not allowed to use calls to functions in this project. However, You may compare your output to the output generated by the image processing toolboxes to ensure you are on the right track. You should write the necessary information in the README file, including the environment, the cmd, the file format, etc. You should make the code in the submission self-contained. The script output should be matched with the file in the results folder.

Cite the paper, GitHub repo, or code url if you use or reference the code online. Please keep academic integrity; plagiarism is not tolerated in this course.