CS 6384 Computer Vision Homework 3

Professor Yu Xiang

March 2, 2022

Download the homework3_programming.zip file from eLearning, Assignments, Homework 3. Finish the following programming problems and submit your scripts to eLearning. You can zip all the data and files for submission. Our TA will run your scripts to verify them.

Install the Python packages needed by

- pip install -r requirement.txt

Here are some useful resources:

- Python basics `https://pythonbasics.org/`
- Numpy `https://numpy.org/doc/stable/user/basics.html`
- OpenCV `https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html`

Problem 1

(4 points) Epipolar Geometry.

Implement the compute_fundamental_matrix() function in epipolar.py. This script first samples a set of pixels on image 1, and then uses the depth and camera poses to find the correspondences of these pixels on image 2. You need to use your backproject function and the method for finding correspondences in homework 1 in this script.

After generating the correspondences, implement the 8-point algorithm to compute the fundamental matrix between these two images. Then the code will sample 3 pixels on image 1 and draw their epipolar lines on image 2 using the computed fundamental matrix.

After your implementation, run the epipolar.py in Python to verify it. Figure 1 shows an example of running the script.

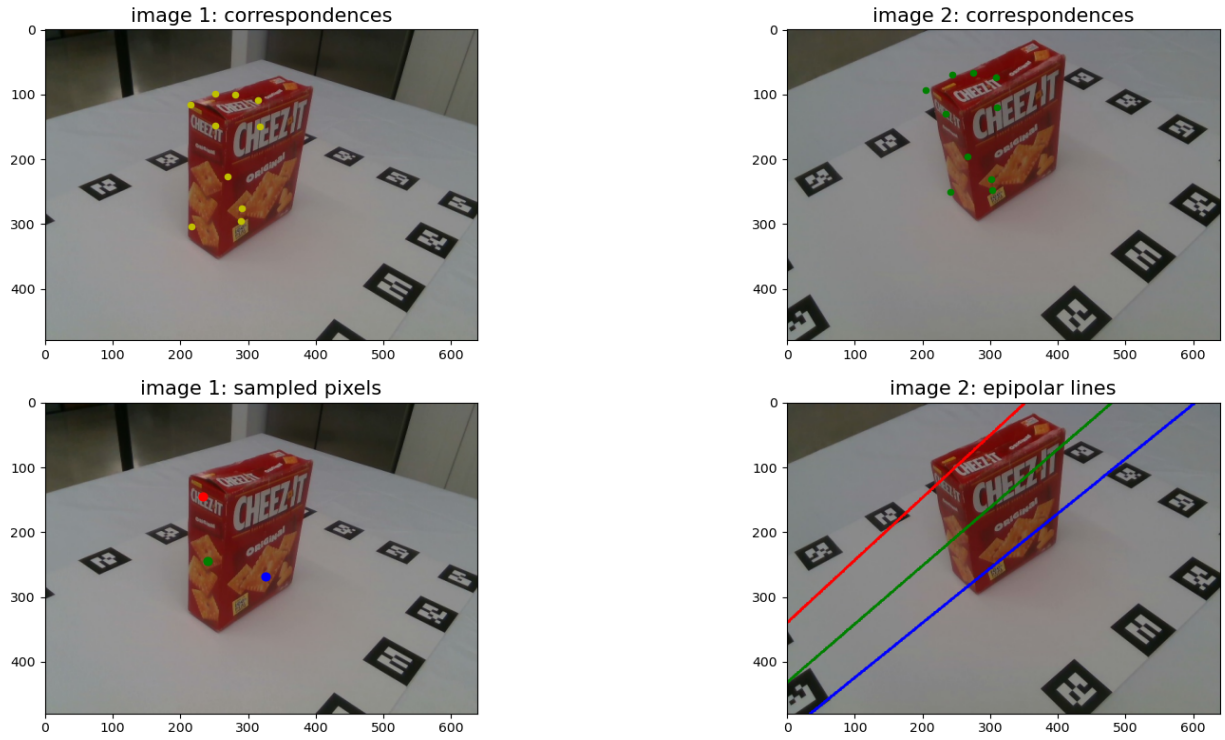

Figure 1: Epipolar lines from fundamental matrix.

Problem 2

(3 points) Triangulation.

Implement the triangulation() function in triangulation.py. This script first samples a set of pixels on image 1, and then uses the depth and camera poses to find the correspondences of these pixels on image 2. You need to use your backproject function and the method for finding correspondences in homework 1 in this script.

After generating the correspondences, implement the triangulation algorithm using nonlinear least squares optimization. The algorithm should output a 3D point for each pair of pixels.

After your implementation, run the triangulation.py in Python to verify it. Figure 2 shows an example of running the script.

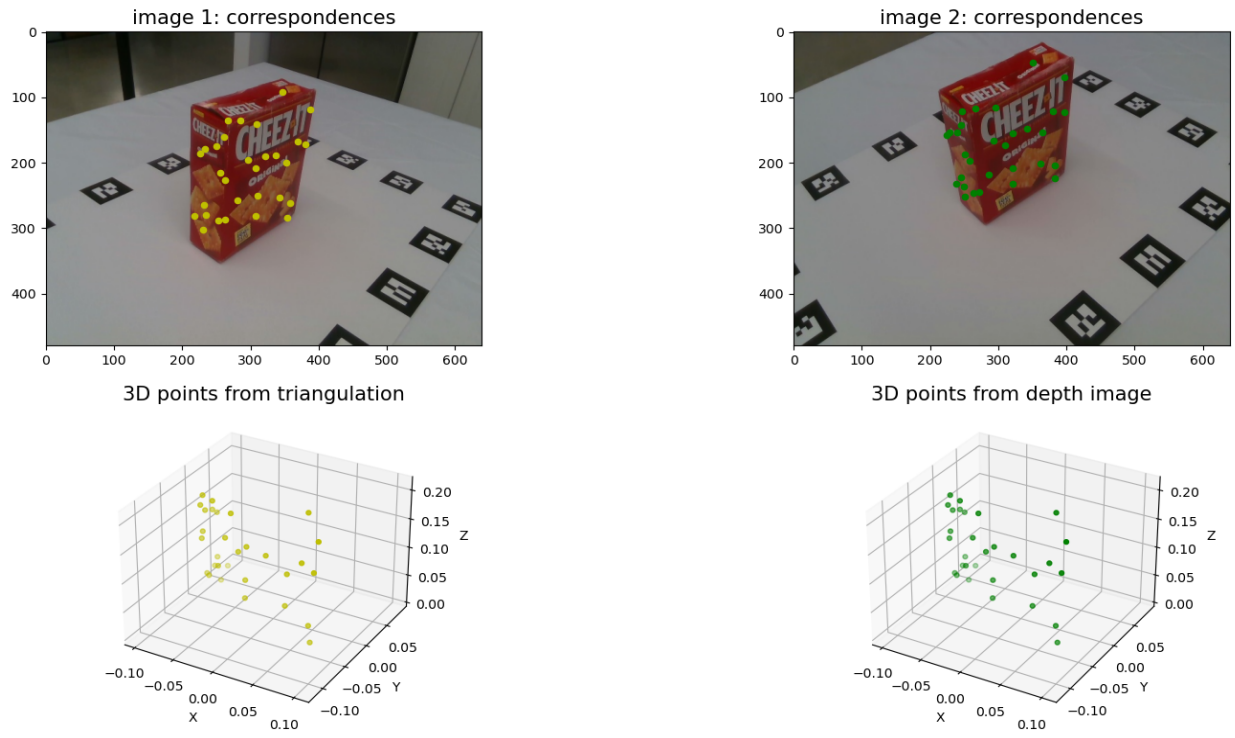

Figure 2: Triangulation to generate 3D points using pixel correspondences with known camera poses of two images. The triangulated 3D points should be similar to the computed 3D points using the depth image and the camera pose.

Problem 3

(3 points) Space Carving.

Implement the space_carving() function in space_carving.py. This function uses camera poses and segmentation masks of two images to reconstruct the 3D shape of the object in terms of voxels with the space carving algorithm.

After your implementation, run the space_carving.py in Python to verify it. Figure 3 shows an example of running the script.

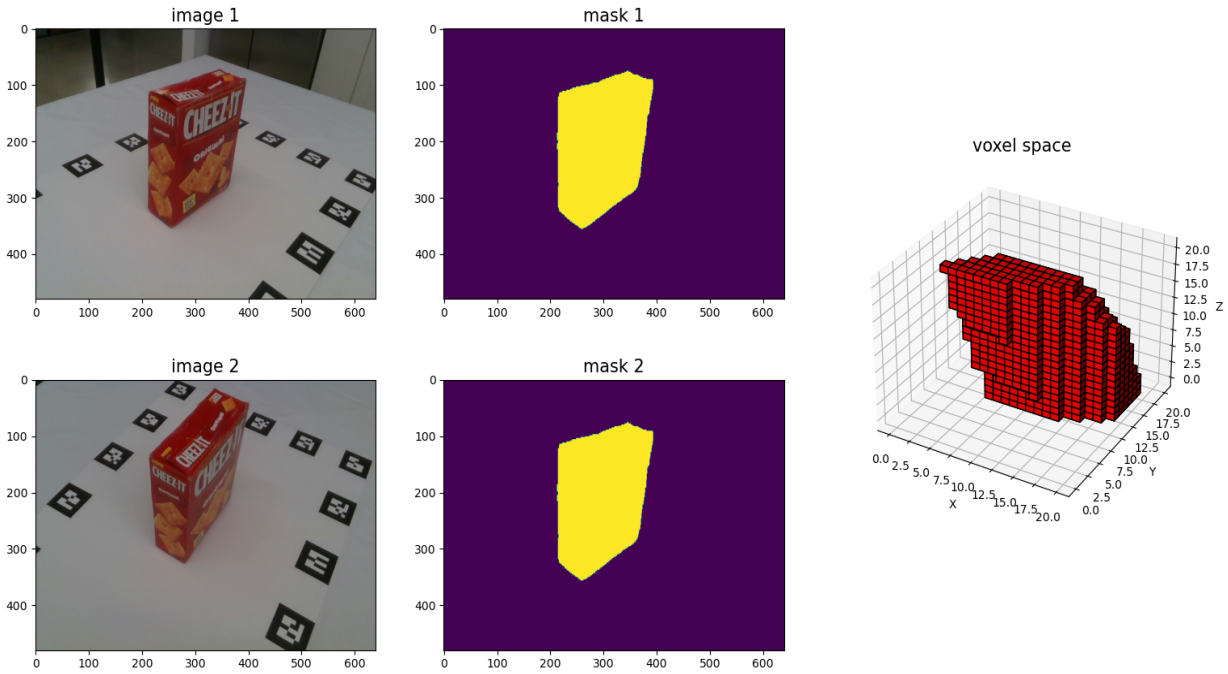

Figure 3: Space carving for 3D reconstruction using two images with their masks and known camera poses. Since we only used two images, the reconstructed shape is very coarse. For example, we do not see the back side of the cracker box.