CS 6384 Computer Vision Homework 2

Professor Yu Xiang

February 15, 2023

Download the homework2_programming.zip file from eLearning, Assignments, Homework 2. 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:

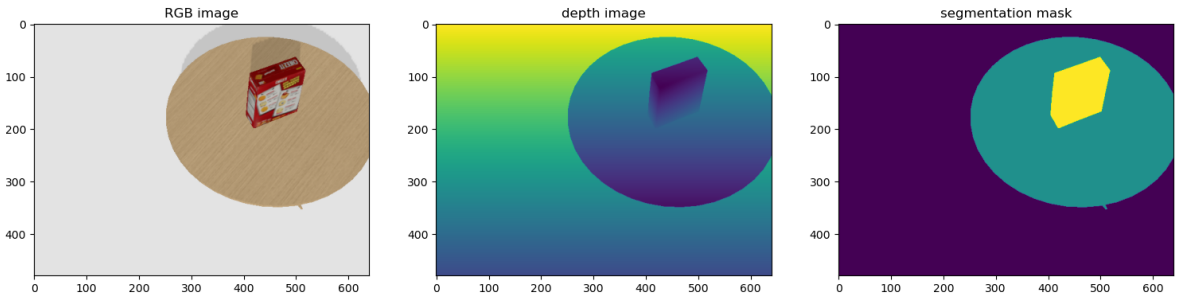
- Python basics `https://pythonbasics.org/`
- Numpy `https://numpy.org/doc/stable/user/basics.html`
- OpenCV `https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html`
- PyBullet `https://pybullet.org/wordpress/`

Problem 1

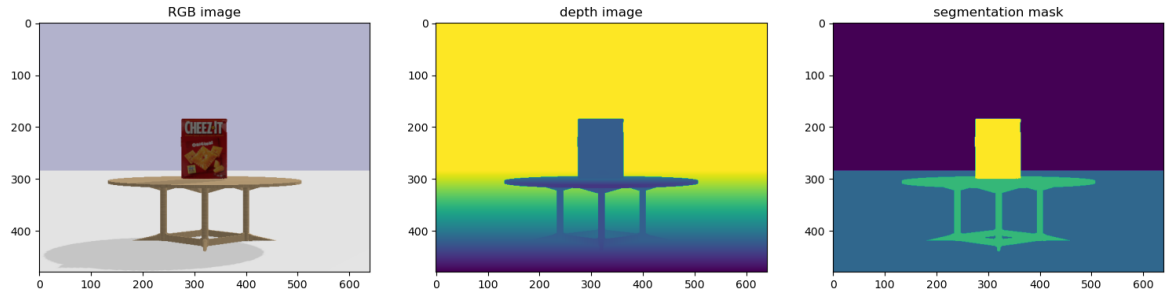
(3 points) Visual Rendering with PyBullet.

Implement the look_at_box_front() function in table_scene.py. This script first builds a 3D scene with a table and a cracker box in PyBullet (a physics simulator). Then it renders images of the scene. The main function first renders an image, then it calls the look_at_box_front() function to change the camera viewpoint to see the front of the box. The script shows the two plots in Figure 1(a)(b) if it runs successfully. You can press "q" to exit the first matplotlib plot window to see the next one.

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



(a) Before changing camera viewpoint



(b) After changing camera viewpoint

Figure 1: Rendering from PyBullet

Problem 2

(4 points) Harris Corner Detector.

Implement the harris_corner() function and the non_maximum_suppression() function in harris_corner.py. The script implements the Harris corner detection algorithm. Follow the steps in the script to implement it.

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

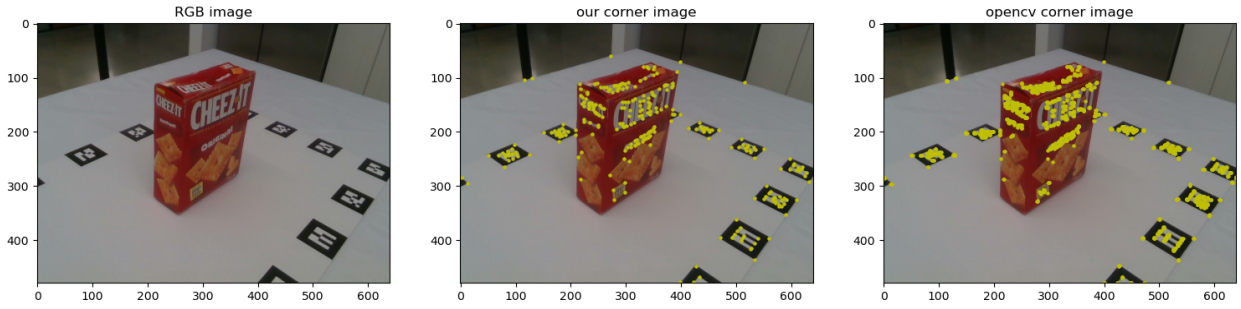


Figure 2: (Left) An input image. (Middle) Harris corner detection with our implementation. (Right) Harris corner detection using the provided function in OpenCV. The yellow dots indicate the detected corners.

Problem 3

(3 points) SIFT feature matching.

Implement the sift_matching() function in sift_matching.py. This script first extracts the SIFT keypoints and descriptors from two images with OpenCV. Then it calls the sift_matching() function to match the detected SIFT keypoints from the two images.

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

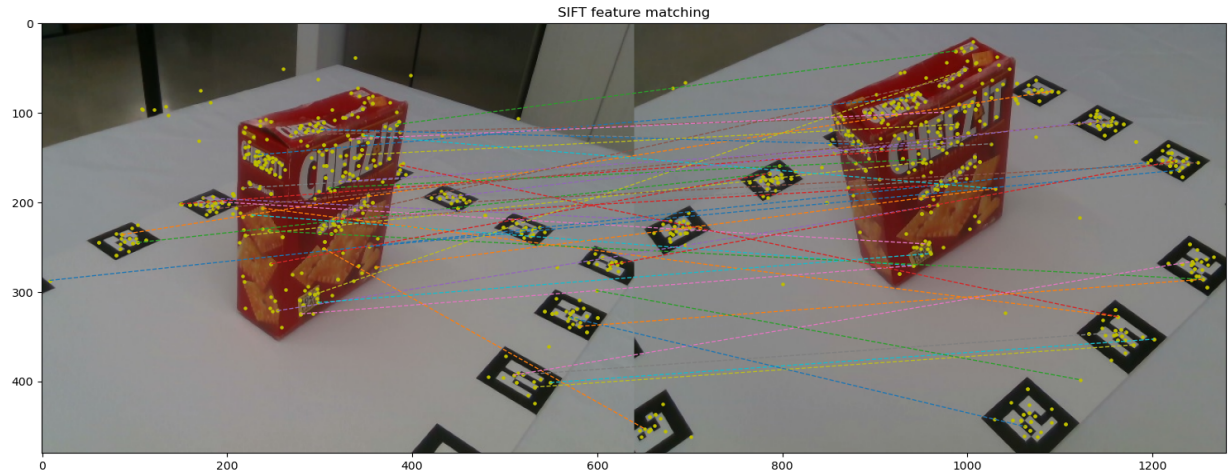


Figure 3: SIFT feature matching between two images. The yellow dots indicate the detected SIFT keypoints and the lines show the matching.