CS6550 Computer Vision
Homework #2

Due: 11:59pm, 11/9/2023

1. **Fundamental Matrix Estimation from Point Correspondences: (50%) (Total 4 image)**

   input : image1 , image2 , pt_2D_1.txt , pt_2D_2.txt

   output : normalized_img1 , normalized_img2 ,wo_normalized_img1 , wo_normalized_img2

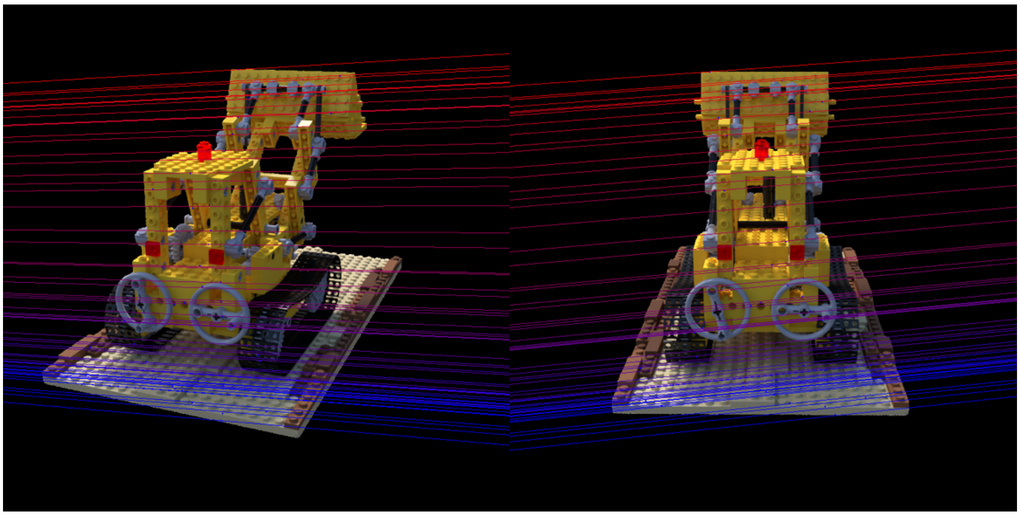
   Note: we provide the matching point pair in `pt_2D_1.txt` and `pt_2D_2.txt` , the first line gives the point number and the others are point coordinate

   In this problem, you will implement both the linear least-squares version of the eight-point algorithm and its normalized version to estimate the fundamental matrices. You will implement the methods and complete the following:

   (a) Implement the linear least-squares eight-point algorithm and report the returned fundamental matrix. Remember to enforce the rank-two constraint for the fundamental matrix via singular value decomposition. Briefly describe your implementation in your written report.

   (b) Implement the normalized eight-point algorithm and report the returned fundamental matrix. Remember to enforce the rank-two constraint for the fundamental matrix via singular value decomposition. Briefly describe your implementation in your written report.

   (c) Plot the epipolar lines with different colors for the given point correspondences determined by the fundamental matrices computed from (a) and (b). Determine the accuracy of the fundamental matrices by computing the average distance between the feature points and their corresponding epipolar lines. Describe your implementation and results.



2. **Homography transform: (50%) (Total 1 image)**

   input : display, post

   output : homography

   You need to determine a homography transformation for plan-to-plane transformation. The homography transformation is determined by a set of point correspondences between the source image and the target image.

   (a) Implement a function that estimates the homography matrix H that maps a set of interest points to a new set of interest points. Describe your implementation.

(b) Specify a set of point correspondences for the source image of the screen and the target one. Compute the 3X3 homography matrix to map the CV image to the screen and using bi-linear interpolation . Please select four corresponding straight lines to compute the homograph matrix. Describe your implementation and show the selected correspondence line pairs, the homography matrix, and the resulting image with the mapped screen region.

Note: using homography matrix mapping screen pixel to CV image and perform bi-linear interpolation in CV image i.e. inverse mapping.

(c) Compute the vanishing point of two parallel lines (upper and bottom) of the screen in the image, show the vanishing point's coordinate and mark it on the image. Describe your implementation.

display	**CV image**

Note : we list some useful API for numpy for speed up your code , use ```for i in range(x)``` as less as possible. Since this syntex is very slow in python.

1. **np.stack()**
2. **np.arange()**
3. **np.meshgrid()**
4. **np.logical_and()**
5. **np.split()**

Reminder:

- **You should not use any function which can generate the result directly in each steps.**
- Your code should display and output your results so that we can judge if your code works correctly.
- Please compress your **code**, **input images**, **result images** , **README** and **report** in a zip file named HW2_{Student-ID}.zip and upload it to eeclass.
- **README must contain information that how to execute your code.**
- If you encounter any problem, please post your problems/questions **on eeclass.**
- Your report must contain your output the result, code and explain your code step by step.
- The repository format is following:

```
.
├── assets
│   ├── display.jpg
│   ├── image1.jpg
│   ├── image2.jpg
│   ├── post.png
│   ├── pt_2D_1.txt
│   └── pt_2D_2.txt
├── hw2_1.py
├── hw2_2.py
└── output
    ├── homography.png
    ├── normalized_img1.png
    ├── normalized_img2.png
    ├── wo_normalized_img1.png
    └── wo_normalized_img2.png
```