Michael DiCioccio
DroneDeplot Coding Challenge Option 1
Pyhton Camera Pose Estimation

Hi DroneDeploy,

Thank you for the opportunity to complete a coding challenge. I decided on option 1 because I have a lot of Python programming experience, but have never used OpenCV before. I have read about OpenCV and have seen a few videos to know how powerful it is. I spent some time reading documentation and deciding on how to approach the challenge before starting.

I began by installing the most recent OpenCV for the image processing part of the problem, numpy for matrix math and manipulation, matplotlib for plotting the images on a scale for comparison, and a few other libraries. The program was written using Python3 and here is a full list of libraries:

- OpenCV (cv2)
- math
- numpy
- pathlib (Path)
- matplotlib (pyplot)

The approach I took to solve the problem was using the original **pattern.png** to find the pattern in the iPhone 6 image of the pattern using OpenCV's SIFT (similar to ORB, SURF, etc.). Then taking the Descriptors from each image I find matches using the Flann matching method. The matches will be iterated through and checked based off a threshold and the values that meet the threshold will be stored in a "good matches" array.

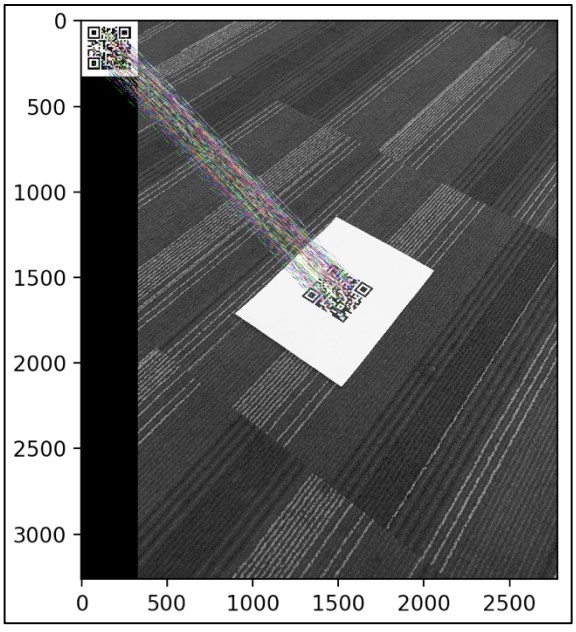

Figure 1: Camera KeyPoint Matching (Pattern to Camera Image)

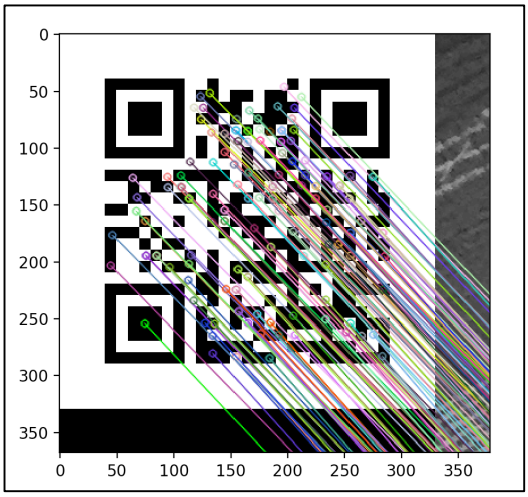

Figure 2: Pattern KeyPoint Matches

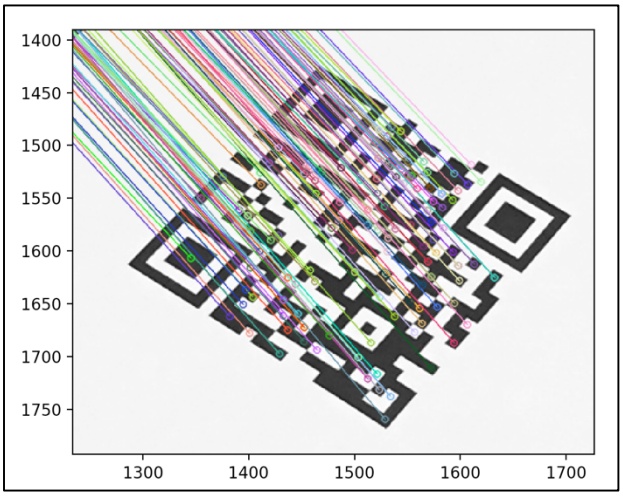

Figure 3: Camera KeyPoint Matches

Now that the QR code pattern was found in the image, the essential matrix needs to be calculated. This will be used to get an estimated pose. The KeyPoints were used to inside of the findEssentialMat method with a camera calibration matrix. This essential matrix is then used in the recoverPose method. The estimated pose will be a matrix containing a scalar relative to the camera distance, a rotation matrix, and a translation matrix. These are used to calculate the x, y, z and yaw, roll, pitch of the camera based off the QR code pattern KeyPoints.

Once the image scalar, rotation matrix, and translation matrix are obtained, trig and simple math was used to get the rotation matrix into Euler angles (radians) and then into degrees. The translations simply need to be multiplied by the scalar c value.

Issues:

While approaching the problem I faced a few issues. First off, I have never used OpenCV before. I have seen what it is capable of in a few videos, but never used or seen any code. I spent a lot of time researching the library and trying figure out the best way to use this library to solve the problem. I still do not think my solution is perfect, and my main reasoning for this is I could not figure out what the camera calibration matrix should have been. Here is the camera matrix I am talking about:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 4: Camera Calibration Matrix

The camera calibration matrix was necessary for the findEssentialMat method. I understand that the focal width is f_x and the focal height is f_y. Also, c_x and c_y are the focal centers of the image. Adjusting these values based off the images being used was difficult and it gave me varying rotations and translation values.

Example Input:

```
Enter the image file path to find the camera pose for: Camera_localization/posed_pattern_1.jpg
```

When the **find_posed_image_data.py** is run a user input is asked. Here is where you will type in the path to the image you want to find the camera pose for. Once it begins running it checks for the **pattern.png** in the same directory, so make sure all the images are still packaged together. I also have code written (commented) to convert the pattern.png into a jpg but it seemed more accurate when using a png.

Resources:

http://16720.courses.cs.cmu.edu/lec/transformations.pdf

https://docs.opencv.org/3.3.0/dc/dc3/tutorial_py_matcher.html

https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#decomposehomographymat

http://www.staff.city.ac.uk/~sbbh653/publications/euler.pdf