Naveen Tumkur Ramesh Babu,
The Ohio State University, USA

Drone Deploy Project Report

Source Code:

Below is the link to my github with solution for option 1 :

https://github.com/naveentrtumkur/Drone_Project.git

Approach Explanation:

My approach can be broken down into following steps:

(1) Finding the key points and finding the match between reference and actual image.

The key points in the image are extracted, by using the sift inbuilt function in opencv. Match between the two images is obtained using brute force matcher (inbuilt function).

(2) Finding the homography and obtaining the matches between images.

Homography helps in picking the inliners from the set of the matches between two images, followed by finding the pattern using the best match(inliners from homography).

(3) Finding the position of camera using the Camera Matrix.

Camera matrix is obtained by mask i.e good match from homography and position of the pattern image.

(4) Plot the camera position on an image.

Finally using the camera matrix values, the camera position is plotted on an image.

Hurdles and Learning Path:

I was very much excited to take this Coding challenge. With little prior knowledge in image processing and python, I felt Challenge 1 could be completed within deadline.

It was great learning understanding basics of openCV, numpy, matplotlib. The rich set of APIs they support and how we can efficiently use python to solve a **real-world Engineering problem**.

Installing Xcode, OpenCv on my MAC was the biggest challenge. There was no directly downloadable installer file available for OpenC Installation. Moreover, my system had lower built-in version of python. I upgraded python from 2.7 to 3.6. I created a virtual environment where I could install my required packages like openCV, numpy, scipy etc for solving this challenge.
Installing OpenCV was biggest challenge. I had to clone source code from a repository, build the source code making relevant changes in cmake file and then install the package. There were various PATH, library mismatch issues encountered and did research to debug them.

Having the working environment ready, then I started learning the required openCV and numpy concepts to come up with an approach to solve my challenge. I started studying concepts on block matching, homography, perspective transforms. I was able to use homography to generate a plot using the Camera Matrix.

REFERENCES:

1. For installing openCv and Python 3.6 on my MAC OS
   https://www.pyimagesearch.com/2016/12/05/macos-install-opencv-3-and-python-3-5/

2. Basic Homography
   https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html

3. Learning Homography
   https://www.learnopencv.com/homography-examples-using-opencv-python-c/

4. NumPy Tutorials
   http://scipy.github.io/old-wiki/pages/Tentative_NumPy_Tutorial