**Report**

**CODEBASE**:

Please find the python program for option 1 : <https://github.com/kruthikavishwanath/dronedeploy>

**EXPLAINATION**:

There are three main parts in this code.

* **Finding the key points and finding the match between two images**

Key points in the image is obtained, by using the sift inbuilt function in opencv.

Match between the two images is obtained using inbuilt brute force matcher.

* **Finding the homography and obtaining the perspective transform**

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

* **Finding the camera position**

Camera matrix is obtained by mask (inliners or good matches) from homography and the position of the pattern image.

* **Plot the camera position on the image**

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

**MY EXPERIENCE :**

I thoroughly enjoyed doing this assignment. Previously, I had very less (less than 6 months) experience of working with opencv c++ , not even opencv python for that matter. As a result, I was a bit nervous when I started this assignment. But, as days went by I developed the attitude to get this done no matter what. I spent nearly 2-3 days on installing opencv python on my laptop. I also, did have tough time figuring the concepts needed before I start coding option 1. But thanks to my friend’s whose MS Thesis report(<http://plan.geomatics.ucalgary.ca/papers/ucalgary_2014_dawar_neha.pdf>) helped me get head start to some of the concepts on perspective transform, optical flow, block matching, and homography. I tried block matching, but failed in it.

I continued my effort with optical flow. Thanks to online resources. It helped me get better perspective in using sift & homography functions in opencv. I enjoyed the most ,while I learnt how a camera matrix was obtained by dot products of the matches and images and also how the camera matrix is transformed to a plot.

**FUTURE WORK:**

Post this submission, I plan to update my old blog (http://django-learnings.blogspot.com/) with “How to successfully install opencv python on mac ?”

I believe I could have done much better job if I had not lost many days installing opencv python. If I had more time, drawing the actual camera in place of plot is something I would have done. I believe some camera plots are not accurate. May be I would have worked on getting better accuracy in positioning the plots.

**REFERENCE**:

* <http://opencv-pythontutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html>
* <http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html>
* <http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html>
* <ftp://ftp.pmel.noaa.gov/vents/Buck/NWROTA_2010/bubble_plume/python_vision.pdf>
* <http://plan.geomatics.ucalgary.ca/papers/ucalgary_2014_dawar_neha.pdf>