Project2 – Task1 Report

Name: K Naga Kiran Reddy Student ID: 50432242

OpenCV Version: 3.4.2.17

Python Version: 3.6.13

pip install opencv-python==3.4.2.17 opencv-contrib-python==3.4.2.17

Process:

1. Loaded images: **img1**-rightimage, **img2**-leftimage

2. Retrieved Key points and Descriptors using **SIFT detector and SIFT descriptor**

3. Implemented **KNN** using a for loop: Variable's definition
   rnorm: stores the norm values between each right and every left key point
   sortedrnorm: stores the sorted norm values in ascending order
   twobestr: stores the Min distance1, Min distance2, index of mindistance1 keypoint, index of mindistance2 key point

4. Used **Ratio testing** to match key points: Taken **n0 = 0.75**
   Stored right and their corresponding left key points in **matchedright** and **matchedleft**
   Retrieved **x and y coordinates** of key points using **kpr[m].pt and kpl[m].pt**

5. Appended 1 to x and y coordinates for easy H matrix calculations

6. Implemented **RANSAC** to calculate **maximum inliers count** [took n = 4 random key point pairs, **t = 5, k = 5000**]

7. We need **4 points** to calculate **Homography Matrix**, so used np.random.choice for randomly picking 4 key point pairs

8. Used **rightv** and **leftv** to store rows of matrix A

9. Retrieved **last row of vtranspose after SVD**, which is the **H matrix**

10. Tested **residual distance** on key point pairs excluding the 4 random points

    This condition is to count the inliers for every iteration.

11. **Maximum inliers count and inliers** are updated on every iteration of K

12. **Largest Inliers set** is used to calculate the H matrix using the SVD determined above.

13. right image and left image corners are calculated

14. Used **perspective transform** to transform right image using H matrix and new corners of right image are calculated

15. Left image points and new right images points are concatenated into **pts**

16. **Xmin and Ymin** values of **pts** are calculated, and

    Translation matrix: **[[1 0 x] [0 1 y] [0 0 1]]** is calculated where x, y is **-xmin and -ymin**

17. **Right image** is wrapped using new matrix (i.e., **np.dot (translation matrix, Homography matrix)**.

18. Left image is stitched to right image by pasting leftimage on to final **result_img**