Project 2

Report – Task 1

Name: Sai Mahesh Pullagura
UB ID: 50419368

OpenCV Version -- 3.4.2.17

Python Version -- 3.6.7 64-bit

Image Stitching:

There are 4 steps performed for stitching 2 images together. Each steps is describes as below.

1. **Keypoints:**
   - The function **"**sift.detectAndCompute()**" is used to** determine the key points and description in both of the images.

2. **Match Keypoints:**
   - In order to obtain the matching key points, we make use of **KNN algorithm** and it is implemented on the **descriptor** points of both images.
   - The KNN algorithm is used to find top 2 best matches. Here we find top 2 pixels in left image that match closely with each of the pixel in the right image.
   - As part of this activity, the function **np.linalg.norm()** is used to find the L2-distance between each of the pixel in right image with all pixels in the left image. The obtained distances are then sorted in non-decreasing order and the top 2 values are stored which are the top 2 best matches for the current pixel in the iteration.
   - Next, to make the list of best matches more refined, we perform **Ratio Testing**. It is controlled with the help of a parameter "η" such that:

     If key_pts[i][0] < η *key_pts[i][1] then store the value, else discard.

   - Now we estimate the actual matching pixel points from the above obtained results using below operations:

     left_img_pts = np.float32([kp_left[k].pt for k in matching_pts])

     right_img_pts = np.float32([kp_right[k].pt for k in matching_pts_ind])

3. Estimate Homography:

- We utilize the functionality of **SVD** algorithm to get the Homography matrix H.
- For this purpose, we randomly choose a set of **n** pixels (=4) from the right image (matching pixels) and use them to implement SVD. Note that since the shape of the Homography matrix is (3*3), the output of SVD will be a vector of shape (9*1). SVD is implemented as follows :

  np.linalg.svd(A)
  H = Vt[8, :]
  H = np.reshape(H, (3, 3))

- Using the Homography matrix, we determine the target matrix which are the calculated left image pixels using the right image pixels. Please note that we only consider the pixels that are not part of the above randomly selected pixels.
- If the difference between the calculated value and the actual value turns out to be less than a value say **t(=**5), then the calculated values are considered to be inliers with respect to **RANSAC** implementation.
- Since we are randomly selecting the pixel values, there is a chance that the selected pixels may not fit the RANSAC model as expected. Hence we run the above steps iteratively (5000 epochs) and update the inlier count and inlier list for each random selection, till we get the best match which fits the model accurately.
- Once the above iterations run successfully, we get the best set of inliers that fit the model as expected. Using these values, we then calculate the final Homography matrix H.

4. Warp and Stitch Images:

- Once we obtain the Homography matrix, then we perform the image warping and stitching operations using the functions **perspectiveTransform()** and **warpPerspective()** respectively.
- Before warping the images, we estimate the maximum and minimum x and y co-ordinates in both of the images which is used to translate the image to the desired position.
- Once the warping process is completed, the final result is returned which is the stitched or the panorama image.

Results:

A sample output for implementing the above process is as follows.