Computer Vision Lab 4

Homography and Image Stitching

In this and next lab, you will practice how to find homography using RANSAC and SIFT given two images. And then you will practice image stitching by transform one image according to homography and stitch the transformed image with the other image. For simplicity we simply blend the overlapping regions using average. The images are shown in the following. For simplicity, you can work with gray images only by converting the input into grayscale using OpenCV.

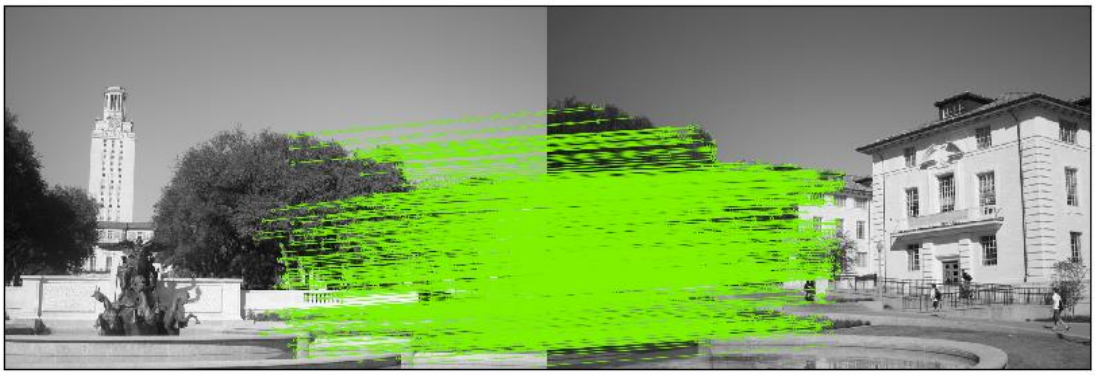
1. **Find Matching Points and Homography** Given above images, try to find homography based on SIFT and RANSAC algorithms. First, use OpenCV functions from last lab to detect SIFT points for each of the images. Then you can use cv2.FlannBasedMatcher() to find matching points among those found in two images. FlannBasedMatcher is a KNN based algorithms for finding points that are close to each other. You may implement your own matching algorithms if you prefer. Then you can use cv2.findHomography() to estimate homograph based on the found matches using RANSAC

   ```
   M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
   ```

   Note that src_pts and dst_pts are the matching points from the two images. M will be the homography. You can visualize all the inliers (for the computed homography) by drawing matches using cv2.drawMatches() function, see following.

   ```
   matchesMask = mask.ravel.tolist()
   draw_params = dict(matchColor = (0,255,0), # draw matches in
   ```

```
green color
                singlePointColor = None,
                matchesMask = matchesMask, # draw only inliers
                flags = 2)
img3 =
cv2.drawMatches(img1,kp1,img2,kp2,good,None,**draw_params)
```



For more details of how to do this task, see https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html#py-feature-homography for reference.

2. **Image Stitching** We then need to stitch the two images according to the homography. The homography computed above should be mapping the left image to the coordinate in the right image. However, when we stitch the two images in to one. The right image should move to the right side of the final image. Hence the homography for transforming the left image into the final image is different from the one computed above. The trick to solve this problem is that for each matched points of the right image, we add the translation from right image to the final image. Let say we assume the width and height of the final image is 5/3 and 4/3 of the original image. We want the right image is placed to the bottom right of the final image. We first translate all the matched points of the right image with 2/3*width in x and 1/3*height in y direction. Then we compute the homography using the new set of coordinates of the right image. We then can use cv2. warpPerspective() to warp the image as follows.

```
img1_warp = cv2.warpPerspective(img1, M, (w+offset_x,
h+offset_y))
```

Note that img1 is the left image. M is the new homography, (w+offset_x,

h+offset_y) is the size of the final image (offset_x is 2/3*width, offset_y is 1/3*height in our example). After warp the left image, you can add the translated right image to the final image. For the overlapping region (pixels with values from both warped left image and right image), you can simply using the average value as the final value. The final stitched image should be like the following.