Computer Vision
Assignment 4

Pranjali Jain
16110119
June 28, 2020

Stereo image correspondences using Fundamental matrix.

1. **Dataset**
    - Captured/Downloaded images of 5 scenes with 2 images each.
    - In a given scene, both images are stereo pairs of each other.
    - The images captured are from stationary scenes with no moving objects.

2. **Grayscaling and Resizing:**
    - All the images are converted to grayscale.
    - The images in each scene are resized only if their size is too big while keeping the aspect ratio same.

3. **SIFT Feature extraction**
    - SIFT and ORB feature extractors available in python-opencv are used to find keypoints and corresponding descriptors in all the images.
    - Keypoints and descriptors help in finding correspondence between two images.
    - SIFT provides descriptors of size 128.

Now, we describe two ways for image stitching: First by implementing from scratch and second using inbuilt functions.

Implementation from Scratch

4. **Feature Matching**

- Between two images, we find the corresponding keypoints using euclidean distances or L2 norm.
- Distance between all keypoints is found using "scipy.spatial.distance.cdist", and then two keypoints from the second image with minimum distance from a given keypoint in the first image are selected.
- We compare the distances of these two keypoints from second image and select only those keypoints where there is a large difference between the difference in the keypoints distance between the keypoint with the minimum distance and the keypoint with the second minimum distance.
- Now we have point correspondences from first image and second image.

5. **Using RANSAC to find fundamental matrix**
   - **Fundamental Matrix estimation**

     In order to find fundaamental matrix, we randomly sample eight points from the point correspondences obtained using feature matching.

     Using **Eight-point Algorithm**,

     We have, $\mathbf{Af = 0}$ we find the $\mathbf{f}$ matrix.
     $\mathbf{f}$ is the last row of $\mathbf{V}$ in the SVD of A, $\mathbf{A=UDV^T}$
     We reshape $\mathbf{f}$ to obtain $\mathbf{F}$
     $\mathbf{F}$ is a 3x3 matrix

   - **Finding Inliers**

     We find inliers by using the epipolar constraint $\mathbf{X_R{}^T F\, X_L = 0}$. The points that lie in a threshold are taken as inliers.
     Here $\mathbf{X_R}$ are the points in the source image and $\mathbf{X_L}$ are the points in the reference image.

   - **Running RANSAC**

     We run ransac algorithm on the obtained $\mathbf{F}$ matrix and inliers, and prune the results. We run the algorithm till the size of inliers is found to be more than a threshold value(0.8*number of point correspondences).

6. **Finding Epiliines in source image using points from reference image**
   - After estimating the $\mathbf{F}$ matrix, we find the Epilines in source image using points from reference image.
     $\mathbf{L_R = F\, X_L}$

Using inbuilt functions

7. **Feature Matching**
   - Between two images, we find the corresponding keypoints using euclidean distances or L2 norm.
   - We do this using **BFMatcher** class in cv2, and **knnMatch** function. This works in a similar way to the implementation from scratch described above.
   - We find the point correspondences between two images that have the minimum distance from each other.

8. **Using RANSAC to find fundamental matrix**
   - **Fundamental Matrix estimation**
     In order to find fundaamental matrix, we use the **findFundamentalMatrix** function.

   - **Finding Inliers**
     We find inliers by using the epipolar constraint $\mathbf{X_R}^{\mathbf{T}}\mathbf{F}\,\mathbf{X_L} = \mathbf{0}$. The points that lie in a threshold are taken as inliers.
     Here $\mathbf{X_R}$ are the points in the source image and $\mathbf{X_L}$ are the points in the reference image.

   - **Running RANSAC**
     We run ransac algorithm on the obtained **F** matrix and inliers, and prune the results. We run the algorithm till the size of inliers is found to be more than a threshold value(0.8*number of point correspondences).

9. **Finding Epiliines in source image using points from reference image**
   - After estimating the **F** matrix, we find the Epilines in source image using points from reference image. This is done using the **computeCorrespondEpilines** function.

The following steps are similar for both implementation from scratch and inbuilt functions.

10. **Finding points corresponding to epilines in source image**
    - We choose a particular value of x and find all corresponding points on the source image that lie on the epiline.

11. **Finding descriptors of all points in source image and reference image**

- Using SIFt, we find descriptors for all points in source and reference image.

12. **Comparison of descriptors from source image and the points obtained from epilines that lie on source image**
    - The descriptors of points obtained from epilines are compared with points on the source image. We find the closest point to a given point in source image.
    - We create mappings of these points. The mapping is from a given point in reference image to a point in source image.

13. **Creatin new image using point mappings from reference image**
    - Finally we create a canvas big enough to hold the new image.
    - We place the source image patches on the canvas using the mapping obtained in the previous step.
    - The canvas now shows the new image which is geometrically similar to the reference image made using the image patches of the source image.

Sample images from the dataset and results

Scene 1

Reference image **Source image**

Implementation from scratch

Inbuilt implementation

Insight : The results obtained using the implementation from scratch are not very good. This is because a precise fundamental matrix is not getting calculated.

References:

1. http://www.cs.cmu.edu/~16385/s17/Slides/12.4_8Point_Algorithm.pdf
2. http://www.cse.psu.edu/~rtc12/CSE486/lecture20_6pp.pdf
3. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MOHR_TRIGGS/node50.html
4. https://www.cs.unc.edu/~marc/tutorial/node54.html
5. https://docs.opencv.org/master/da/de9/tutorial_py_epipolar_geometry.html
6. https://medium.com/@dc.aihub/3d-reconstruction-with-stereo-images-part-3-epipolar-geometry-98b75e40f59d
7. http://ros-developer.com/2018/12/21/computing-fundamental-matrix-and-drawing-epipolar-lines-for-stereo-vision-cameras-in-opencv/
8. https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf
9.