Study and Implementation of Panorama Stitching

Danish Gada, Sushant Kumar, Raj Ghugare
Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur, India
danishgada2905@gmail.com
sushantku24@gmail.com
rajghugare198@gmail.com

***Abstract*—Capturing images is an age problem open for research. Researchers have often used various methods to improve the capturing of images. Research is being carried out in various fields from developing hardware to improving software that enhance picture quality. This is a vast field and has seen immense growth in recent years. There are a lot of open ended problem statements in the field of computer vision and image processing. One of the problem statements are to capture images wider than the camera lens. This is done by taking multiple images and stitching them to obtain a single panorama image. In this paper we use different methods to perform image stitching and obtain panorama images and show the results.**

***Index Terms*—Panorama Image, Harris Corners, Image Stitching, SIFT, OpenCV**

I. Introduction

The concept of Image Stitching has been around for quite some time and has been one of the most successful implementations of Computer Vision. These days, panoramas can be shot on every smartphone. In this project, we study and implement some robust and well-establised "Feature Matching" techniques to build a real-time Image Stichting Algorithm.
We shall discuss each of the algorithms in brief. The coding here is done in Python-3, using the Numpy framework for optimizing.

A. Steps involved in Harris Corner Detection

In Harris corner detection It we basically find the difference in intensity for a displacement of (u,v) in all directions. This is expressed as below:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

Window function is either a rectangular window or Gaussian window which gives weights to pixels underneath.
So we Get Two Sobel Derivatives SobelX and SobelY in X and Y direction Respectively.

We have to maximize this function E(u,v) for corner detection.

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum x,y w(x,y) \begin{bmatrix} IxIx & IxIy \\ IxIy & IyIy \end{bmatrix}$$

So in the Next Step we Compute a Hessian Matrix of Size 2. (Named as M) , For which we will need to Calculate Ix * Ix , Iy * Iy , Ix *Iy , Iy * Ix where Ix is SobelX and Iy is SobelY.

This, they created a score, basically an equation, which will determine if a window can contain a corner or not.

$$R = det(M) - k(trace(M))^2$$

We apply Gaussian Blur to the Image Now we Will calculate The Value of a Parameter R where

$$det(M) = \lambda_1 \lambda_2 trace(M) = \lambda_1 + \lambda_2$$

Here, λ_1 and λ_2 are the eigen values of M

So the values of these eigenvalues decide whether a region is corner, edge or flat. When is small, which happens when and are small, the region is flat. When , which happens when or vice versa, the region is edge. When is large, which happens when and are large and , the region is a corner.

B. Results

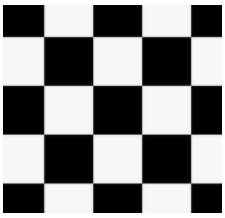

Fig. 1. Checker Board Pattern

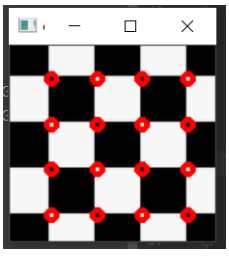

Fig. 2. Checker Board Pattern with Corners detection

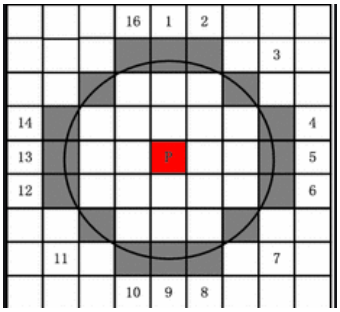

Fig. 3. Target image

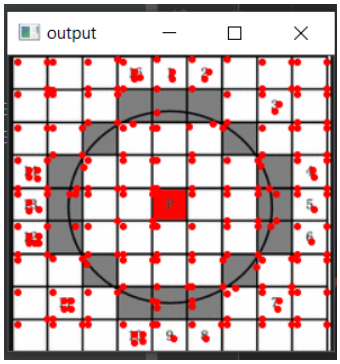

Fig. 4. Target Image with Corners Detected

II. Image Stitching using Homography

- **Feature Detection on images using SIFT**
  SIFT stands for Scale Invariant Feature Transform, which is used for detection features on the images.
  There are mainly four steps involved in SIFT algorithm.

  – **Scale-space Extrema Detection**
    Unlike the Above Harris Corner Detection Method,The Speciality of SIFT is that the key-points Detected by Sift Algorithm are Scale Invariant.
    To achieve this, windows of different scales are taken. This is done using a Method Called Laplation of Gaussian or **LOG** for various values of sigma. This will give us a list of (x,y,sigma) values which means there is a potential key point at (x,y) at sigma scale. But this LoG is a little costly, so SIFT algorithm uses Difference of Gaussian which is an approximation of LoG.
  – **Key point Localization**
    After locating the key-points , Taylor series expansion of scale space is used to get more accurate location of extrema. To do this we use a 2x2 Hessian matrix (H) and compute the principal curvature and apply the knowledge that one Eigen value is larger than the other for edges.
  – **Orientation Assignment**
    gradient magnitude and direction is calculated and a histogram is created. of it is also considered to calculate the orientation. It creates key-points with same location and scale, but different directions. It contribute to stability of matching.
  – **Key point Descriptor**
    Final Step is to create a key point descriptor. A 16x16 neighbourhood around the key point and divided into 16 sub-blocks of 4x4 size.
    For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available.

  The Example Image taken for representing the action of **SIFT** is shown below.

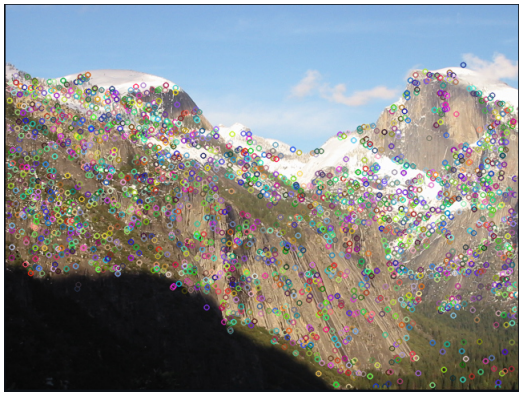

Fig. 5. Test Image

- **Feature Matching between two images using Brute Force K-Nearest Neighbour Matching**
  Here, we take a descriptor of a feature from the first set and match it with the descriptors of all other features from the second set. Distance is calculated using the L2 norm, and the closest one is returned. We chose to remove the cross-checking in our Algorithm to improve its run time. Also, instead of drawing all the best matches, we draw k = 2 best matches, that is, drawing two match-lines for each key-point to improve run time while also making sure that the feature is a Reliable one. Its output with 1.jpeg and 2.jpeg is shown below.[2]

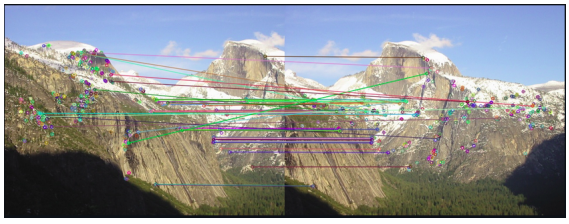

Fig. 6. Feature Matching using Brute Force KNN Matching

- **Generation of the Transformation Matrix.**
  Planar Homography relates the transformation between two planes. It is a 3x3 matrix with 8 Degrees of freedom. The elements of the matrix are generally normalized to 1. Here, To transfer an image from a source plane to a target plane, we determine a Homography matrix and apply it on the source plane, to transfer the image into the target

plane, as per the matched features. In this way, we can obtain the result as shown.[3]

Fig. 7. Generation of the Transformation Matrix

In the code, we did the planar image stitching for multiple images. So, we initially made a Black canvas of 5 times the length of the image, and height 200 pixels greater than the height of the image to account for any errors in the vertical direction. After the first two images have been stitched as shown, Using the third image, we calculate the homography matrix between the third image and the second image that had been warped in the first iteration, before it was stitched to the source image. Similarly, the homography matrix between the fourth image and the third image before wraping is calculated. Stitching of all the four images results as shown,

Fig. 8. Stitching of all the four images Using Planar Stitching

III. Image Stitching using Homography

- **Conversion from Planar to Cylindrical Coordinates**
  Conversion from Planar to Cylindrical Coordinates. Image Stitching in cylindrical coordinates requires the use of a tripod for recording the video, since then the motion along the y axis will be as good as null. In such a scenario, we use the transnational model for image stitching. But before that, we should convert the image into cylindrical coordinates. According to [4], image can be transferred to a cylinder using the formula listed in the slides. When implemented, its results are as shown,
  For the image shown above, the intrinsic camera parameters are found, and used for demonstration of the algorithm

- **Feature Detection on images using SIFT Covered Above**

- **Feature Matching between two images using Brute Force KNN. Covered Above**

Fig. 9. Conversion from Planar to Cylindrical Coordinates

- **Translational Model**
  Now that the feature points are obtained for the pair of images using SIFT, it is time for the translational model for stitching.Here, we have approximated the least mean square technique as used in [6] by taking the difference of the obtained feature points between two consecutive images and take the mean of the result .

  We also find its standard deviation and divide the answer into the standard deviation along x axis and y axis. We then establish a threshold. Only if the value of the standard deviation along both the axes is less than threshold,the perspective warping of the second image with-respect-to the first image shall be performed. This is done for gap closure.

  After the distance between the Keypoints are calculated as stated, we divide the array into the distance along the x axis and along y axis, which is then fed into the translational matrix. This matrix is used for the perspective warping of the images. The array is similarly updated for each iteration. The translational matrix is generated between the next image and the warped version of the previous image. In this manner, we can obtain the cylindrical panorama.

  The length of the canvas is equal to the circumference of the cylinder, that is, 2* pi * f, where f is the focal length of the camera, extracted from the intrinsic camera parameters, and the height being 200 pixels more than the actual height of the image.

Note that conversion from planar to cylindrical coordinates requires the use of the focal length, which can be found using the intrinsic camera parameters. These can be found using camera calibration. To calculate these parameters, we used a chessboard for getting the corner locations of an image.

A. Conclusion

The real - time implementation was performed by recording the video of our lab, where the camera was placed roughly at the centre of the room and rotated at a roughly uniformly angular velocity. The camera used was my phone camera but instead we could have used a depth camera for eg:- using pyrealsense2 for recording the video and for camera calibration. The result for this can be seen below

Fig. 10. Final 360 Cylindrical Panorama Stitching Result

REFERENCES

[1] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94

[2] Jakubovic, Amila Velagic, Jasmin. (2018). Image Feature Matching and Object Detection Using Brute-Force Matchers. 83-86. 10.23919/ELMAR.2018.8534641.

[3] http://www.cse.psu.edu/ rtc12/CSE486/lecture16.pdf

[4] http://cs.brown.edu/courses/cs129/results/final/yunmiao/ .

[5] https://github.com/saurabhkemekar/camera-calibration