Report: Image Stitching

Frank Cally Tabuco

Categories and Subject Descriptors

•**Computing methodologies** → **Image representations;**

Keywords

feature matching, feature detection, harris detector, RANSAC

1. INTRODUCTION

Computer vision follows the same procedures instantenously processed by humans when perceiving a certain scene or image. It involves the acquisition of an image through an eye or lens and interpretation of pixels, shapes, and colours through the brain. The main goal of computer vision is to analyze multiple images and videos to achieve human-level perception or even greater. This task would help in various fields requiring high level of image analysis such as biomedicine[6], physiology[2], and bioacoustics[5].

In this programming assignment, the task is to use the OpenCV libraries[1] to stitch two images together to form a panoramic image. We aim to implement image stitching from scratch which uses Harris corner detector as the feature detector.Additionally, RANSAC algorithm is also used to retrieve a homography mapping of one image into the other. The rest of the paper is structured as follows: section 2 will discuss the techniques and algorithms employed to stitch two images. In section 3, the results of the image stitching program is discussed along with improvements. The last section will discuss the conclusions for this assignment.

2. METHODOLOGY

Two images of a building captured at different angles are shown in Figure 1. The left image is called query image which will be the basis for the transformation of the train image, right picture. The main objective of image stitching is to find points in one image which are the same or similar with another image. From those points the two images would be overlapped to form a panoramic image. We can easily see in Figure 1 that the right half of query image is the same as the left half of the train image but captured at different angles.

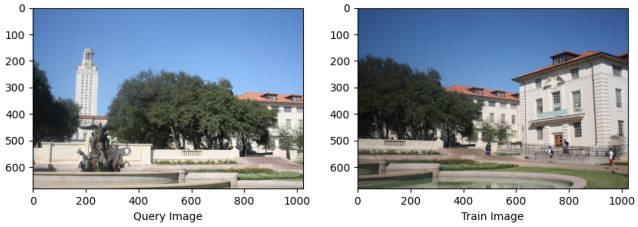

Figure 1: Images to be stitched together.

2.1 Harris Corner Detector

The feature detector is the most crucial algorithm to achieve a stitched image. For this implementation, Harris corner detector is used. The algorithm involves computing for tthe differences in pixel values within a shifting window. Harris and Stephens [4] expressed this intensity displacement as:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \quad (1)$$

The function w(x,y) is the window function and inside the brackets are intensity shifts. They then applied a Taylor expansion to achieve an estimation of E(u,v) given by:

$$E(u,v) \approx [uv]M \begin{bmatrix} u \\ v \end{bmatrix} \quad (2)$$

where

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y y \end{bmatrix} \quad (3)$$

I_x and I_y are sobel derivatives. From there they formulated a score to determine if a window is a corner or not. The score uses the formula below:

$$R = det(M) - k(trace(M))^2 \quad (4)$$

where,
det(M) = $\lambda_1 \lambda 2$
trace(M) = $\lambda_1 + \lambda 2$
λ_1 and λ_2 are the eigen values of matrix M

ACM ISBN XXXXXXXX.
DOI: XXXXXXX

This algorithm is implemented in the OpenCV library as cv.cornerHarris(). The OpenCV function takes input images in grayscale format along with the blocksize or window size, the sobel parameter size, and Harris free parameter. Corners from the grayscale-converted query and train images which satisfies a certain threshold are then stored in an array. For this implementation, corner values must have a value greater than 10% of the max corner value array. Point coordinates of these corners are then retrieved from the original images and stored as query_pts and train_pts for query and train image, respectively. These point coordinates now constitutes the keypoint features of the two images. To easily implement the succeeding algorithms and functions, these point coordinates are converted into keypoint format using the cv.KeyPoint function.

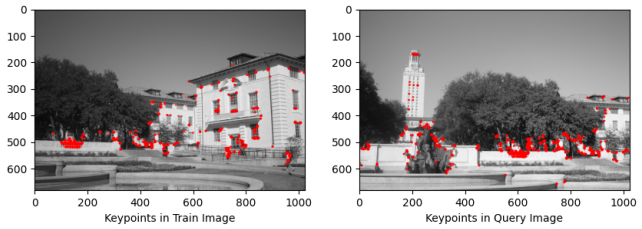

Figure 2: Keypoints in grayscale images.

2.2 Feature Descriptors

To accurately match features in train image from those in query image, we need to have a descriptor for each keypoints. A simple approach to this is centering the keypoint and extracting fixed-size patches around it. After some experimentation, the patch size of 8 is used since no difference in output images when using bigger patches. Bigger patch sizes only produced longer run times without improvement in stitched image quality. The feature descriptor extraction algorithm is shown below:

Table 1: Feature Descriptor Extraction Pseudocode.

```
I → keypoints // set of keypoints in image
Size → 8
Half → 4
row,col → image.shape
For i in I:
    x,y → i //extract x,y coordinates
    if x - Half <0: //out of bounds
        beginX → 0
        endX → beginX + Size
    Else if x + Half > row:
        endX → row
        beginX → endX - Size
    Else:
        beginX → x - Half
        endX → x + Half
    Repeat for y
```

From these patch sizes, patches are then extracted and stored into arrays query_ft and train_ft. These arrays of feature descriptor vectors are then matched against each other by using Euclidean distance. Feature descriptors in query image with a short Euclidean distance in feature descriptors in train image will constitute the putative matches for both images. The BFMatcher function in OpenCV is used to implement matching.

2.3 Affine and Homography Transformations

The computed putative matches are then used to estimate affine and homography transformations from the two images. According to documentations from OpenCV, cv.estimateAffinePartial2D() and cv.findhomography() uses RANSAC as method for estimating the transformations. Random Sample Consensus (RANSAC) is an algorithm introduced by Fischler and Bolles [3]. The idea of RANSAC is to construct a line using two randomly selected points. From this line, supportive points within a certain threshold distance of the line are computed. The line with the most number of inliers constitutes the robust line. A comparison of inliers and residuals for both transformations are shown below.

Table 2: Inliers and residuals.

	Inliers	Ave. Residual
Affine	281	213,857.15
Homography	379	212,218.35

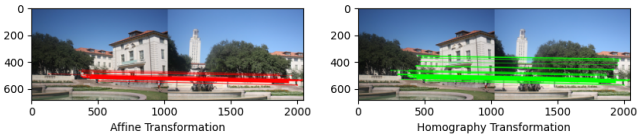

Figure 3: Matches from each transformation

3. RESULTS

To warp the train image based on putative matching points, we need the homography matrix we acquired from the homography transformation procedure. The 3x3 matrix is then used as input in the function cv.warpPerspective(). The resulting image is shown Figure 4. We can see that the train image is pushed back on regions with inlier matches from the query image. All that is left to do is to include the query image in the plotting to form a seemingly stitched image shown in Figure 5.

Figure 4: Warped Train Image

It can be seen from the resultant image that there are points which did not align well. There are two potential reason for this. One, the feature detector technique we employed here is outdated. There are more sophisticated methods which are more robust to invariance. If we look at the original image, the scale of the train image is more magnified

Figure 5: Stitched Output

than the query image. Harris detector, however, is not invariant to scale which resulted to mismatching points in the stitched image . Another reason is the feature descriptor used. A simple square window is used with the keypoint at the center of the window. Employing better feature descriptor techniques could yield better results.

4. CONCLUSION

We have implemented an image stitching program from scratch using Harris corner detection as the keypoint feature extractor. The descriptors for each keypoints are then retrieved using an algorithm which computes for fixed, square sized patches. Using RANSAC to estimate a homography transformation, we warped the train image and overlap the query image from the resulting plot. The resultant image could still be improved using modern techniques such as SIFT. However, for the purpose of implementing an image stitching program from scratch, corner Harris detector is suitable to learn the technicalities in feature matching.

5. REFERENCES

[1] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[2] Cho, Y. Rethinking eye-blink: Assessing task difficulty through physiological representation of spontaneous blinking. *In Proceedings of CHI Conference on Human Factors in Computing Systems (CHI'21)* (May 2021).

[3] Fischler, M. A., and Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (June 1981), 381–395.

[4] Harris, C. G., and Stephens, M. A combined corner and edge detector. In *Alvey Vision Conference* (1988).

[5] Lasseck, M. Acoustic bird detection with deep convolutional neural networks. In *Detection and Classification of Acoustic Scenes and Events 2018* (2018).

[6] Lopez, F., Varela, A., Hinojosa, O., Mendez, M., Trinh, D.-H., ElBeze, J., Hubert, J., Estrade, V., Gonzalez, M., Ochoa, G., and Daul, C. Assessing deep learning methods for the identification of kidney stones in endoscopic images, 2021.