ASSIGNMENT 2

Asvini Selvaraj, Matriculation Number: G2303641A Shwetha Ravi, Matriculation Number: G2304191F

1. ABSTRACT

This report describes the development of an image stitching program that leverages homography techniques. The primary stages of program development include feature detection and matching, robust homography recovery, and the actual process of image stitching to create a panorama. The subsequent sections of the report provide an exhaustive description of the adopted algorithms, design choices, stitching results and potential enhancements.

2. INTRODUCTION

Image stitching is a method of combining numerous photos to create a single panoramic or wide-angle image. It entails smoothly aligning and mixing the photos to produce a cohesive and continuous effect. Feature extraction, feature matching, transformation estimation, and picture mixing are all common processes in the image stitching process.

Each image's distinguishing features are extracted during the feature extraction process. These can be points, corners, or other visual landmarks that are not affected by scale, rotation, or affine transformations. For feature extraction in image stitching, the Scale-Invariant Feature Transform (SIFT) technique [1] is often employed.

Following feature extraction, the next step is feature matching, which identifies similar features between pairs of photos. This is accomplished by comparing the extracted features and matching them based on similarity. For feature matching, many algorithms can be utilized, such as the closest neighbor algorithm or the RANSAC algorithm [1].

After identifying the appropriate features, the transformation between the images is estimated. Finding the geometric transformation that aligns the photos based on the matching features is required. Translation, rotation, and homography are examples of common transformations. The RANSAC technique is frequently used to estimate transformation parameters robustly and reduce outliers [1].

Following the estimation of the transformations, the images are twisted or altered to align them depending on the estimated transformations. This guarantees that the photos' overlapping portions are appropriately aligned. Finally, the matched photos are combined to form a smooth and visually appealing panoramic or wide-angle image. Blending techniques such as feathering, gradient-based blending, and multi-band blending can be applied [2].

Image stitching has numerous uses in industries such as photography, virtual reality, surveillance, and medical imaging. It enables the creation of wide-angle or panoramic photographs with a wider field of view than a single image. The accuracy of the feature extraction, feature matching, and transformation estimation phases, as well as the effectiveness of the blending technique used, determine the quality of the stitched image.

3. DETECT AND MATCH FEATURES

3.1. Feature Point Detection

Feature point detection involves identifying points in an image which are characterized by unique attributes, making them recognizable across varying conditions. These points known as keypoints are typically robust to changes such as rotation, scale, and illumination. Detecting these points accurately is important for the image stitching task.

3.1.1. Methods and Approaches

Notable algorithms for feature point include:

- Harris Corner Detector: This method identifies the corners in an image as feature points. Corners are regions in the image with large variations in intensity in all directions. The Harris Corner Detector works by determining the difference in intensity for a displacement of (u, v) in all directions. [3]
- Difference of Gaussian (DoG): This method involves the subtraction of one Gaussian blurred version of the original image from another, less blurred version. The result is a difference image or DoG, which preserves the edge-like structures in the image and can be used to detect features. [4]

3.1.2. Improving Detection with Adaptive Non-Maximal Suppression [6]

However, the detected feature points can be densely clustered in certain image regions, while other regions remain sparse. This imbalance can be addressed using adaptive non-maximal suppression (ANMS). ANMS ensures a more evenly distributed set of feature points by suppressing "non-maximal"

points in dense regions. This leads to better distribution and also reduces the number of feature points to process in the following stages, thus optimizing performance.

We used ANMS for this task:

1. If the number of input keypoints is less than or equal to the desired number of keypoints to retain, the function simply returns all input keypoints.
2. For each keypoint, its strength or significance is determined by its response value. Note that the code uses the negative of the response to sort in descending order since high response values indicate stronger keypoints.
3. The top keypoints based on strength are selected based and returned.

3.1.3. Standardization of Detected Features

After detecting the keypoints / feature points, it is essential to standardize the features for consistent results across different scenarios. By subtracting the mean and dividing by the standard deviation, each feature vector is normalized. This ensures that the features are scale-invariant.

3.2. Feature Descriptor

A feature descriptor encapsulates the local information or appearance around a detected point in an image in a way that facilitates robust comparison and matching with other points in different images. This representation, often formulated as a vector, captures the essential characteristics of the region surrounding the feature point, ensuring that it remains distinctive and invariant to various transformations

Feature descriptors encapsulate interesting information into numbers and serve as a kind of numerical "fingerprint" that may be used to distinguish one feature from another.

Ideally, this information would be invariant under picture alteration, allowing us to locate the feature even if the image is altered in some way. Following the detection of keypoints, we compute a descriptor for each of them. Descriptors are divided into two types:

- Local Descriptor: It is a concise depiction of a point's immediate surroundings. Local descriptors attempt to mimic shape and appearance only in a local area around a point, making them ideal for representing it in terms of matching.
- Global Descriptor: It is a word that describes the entire image. They are often not particularly resilient because a change in a portion of the picture can cause it to fail because the resulting descriptor is affected.

Main Feature Descriptors are as follows:

1. SIFT: Scale Invariant Feature Transform.
2. SURF: Speeded Up Robust Feature
3. BRISK: Binary Robust Invariant Scalable Keypoints.
4. BRIEF: Binary Robust Independent Elementary Features.
5. ORB: Oriented FAST and Rotated BRIEF.

SIFT and SURF methods attempt to alleviate the limitations of corner detection techniques. Corner detector techniques typically employ a fixed-size kernel to find locations of interest (corners) on pictures. It is clear that when we scale an image, this kernel may become too small or too large.

To overcome this constraint, approaches like as SIFT employ the Difference of Gaussian (DoG). The goal is to use DoG on various scaled copies of the same image. It also makes advantage of nearby pixel information to discover and refine critical locations and descriptors.

When two identical images are present, but one is scaled differently than the other, SIFT maximizes the Difference of Gaussians (DoG) in scale and space to find the same key points independently in each image." DoG is the difference between two Gaussian blurrings of a picture with differing standard deviations. Every octave, or scale, of the image is blurred with Gaussians with varying scaling factor standard deviations. DoG is used to calculate the differences between nearby Gaussian-blurred photos. The method is repeated for each scaled image octave.

3.3. Feature Matcher

3.3.1. Brute-Force Matcher

BF Matcher compares the descriptor of a feature in one picture to all the other features in another image and gives back the match based on how far apart the two images are. Here are two sets of features: those from image 1 and those from image 2. Each feature from set 1 is compared to all the features in set 2. Due to the number of checks, it takes a while.

There is an easy way to figure out which part of the query picture matches the train image the best. The best match is the one with the smallest gap between a feature in the train picture and all the features in the query picture.

The Brute-Force matcher is easy to use. That feature's description is matched with all the other features in the second set by figuring out how far apart they are. The closest one is then sent back.

3.3.2. Flann Matcher

Fast Library for Approximate Nearest Neighbors - FLANN. It includes a number of methods that are tailored for high dimensional features and quick nearest neighbor searches over big datasets. With larger datasets, it performs more quickly than BFMatcher. Figure 2 shows the examples, which uses a matcher based on FLANN.

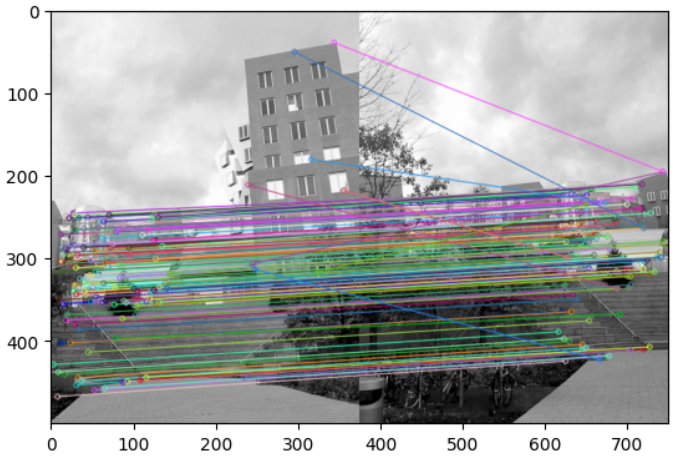

Figure 1

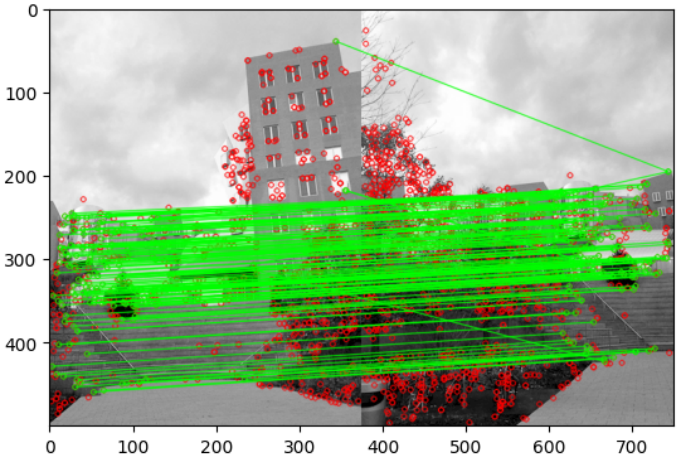

Figure 2

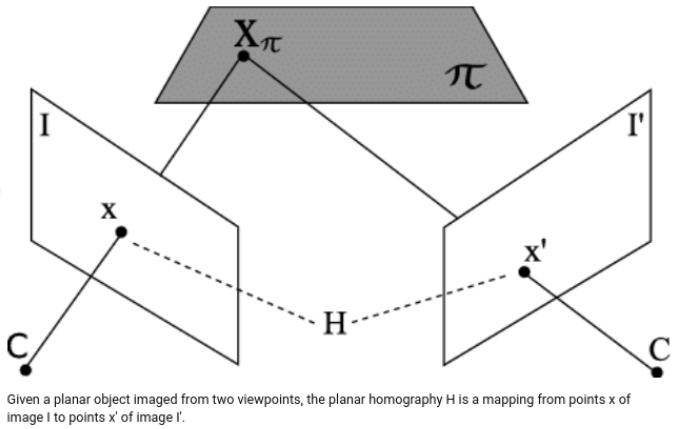

Figure 3

4. ROBUST RECOVERY OF HOMOGRAPHY

With homography, the points in one image are mapped to the points in another image that are the same.

Once we can picture what we're trying to do when we want to put the different pictures together, homography is easy to take in. Think about two situations.

(1) Every time you take a picture, move to the left while keeping the camera still.

(2) You stand in one place and rotate your body while holding a camera. As you spin, you take different pictures.

To put the pictures in (1) together, all we have to do is put one picture on top of the next in the order. But in (2), if we try to join the images together by just putting them on top of each other one after the other in order, we will see that the result is bad because parts of the images will be missed because they were taken on different planes. So, we will need homography to put one picture on top of the other one on the same plane before we can join them. Next, homography can be thought of as a matrix—one that changes points from one image to another image in the same plane. The next question is how to find the homography matrix H.

The homography matrix is 3x3 dimentsional matrix, but it has 8 degrees of freedom.

A homography is a change in the viewpoint of a plane. It means projecting a plane from one camera view into a different camera view, where the camera can move between views and turn.

Perspective transformations turn three-dimensional points into two-dimensional images by using a transformation matrix that includes the focal length, optical center, and extrinsic factors like rotation and translation.

To sum up once more, A homography matrix maps two lines to each other. While we looked at it here, it was a mapping from an image plane to a real plane. However, it could also be a mapping between two image planes. We use projections to connect two images in a homography, which is a type of projective change. People first used homographies to study changes in perspective, and they have helped us learn more about how pictures change when we look at them from a different angle.

When it comes to calculating the Homography matrix between the two images, we can use the below formula:

$$X1 = H * X2 \tag{1}$$

Points X1 and X2 are identical in a separate plane. The way that one point in, say, image 1 matches one point in image 2 is essentially described by the H matrix. The 3x3 homography matrix calculates the conversion between two spaces' collections of 2D points.

4.1. Compute Homography

To find the homography between two pictures, we need to know at least four points that are the same on both of them.

OpenCV predicts a homography that fits all the points that go together in the best way possible. To find the point correspondences, features like SIFT or SURF are matched between the pictures.

To figure out a Homography, the RANSAC method (RANdom SAmple Consensus) is used. It figures out a homography and tells you which pairs are likely to be inliers and which are likely to be outliers. It is also an iterative technique for fitting linear models. Unlike other linear regressors, it is intended to be robust to outliers. It will be used to estimate the Homography matrix in this case. Noting that homography is extremely sensitive to the quality of data we feed it, we require an algorithm (RANSAC) that can filter out unnecessary points from the data distribution.

5. IMAGE STITCHING

We choose SIFT technique as it supports key points as well as keypoint descriptors, where the keypoint descriptor represents the keypoint at a certain scale and rotation with picture gradients. Figure 4 shows the Keypoint of the sample image.

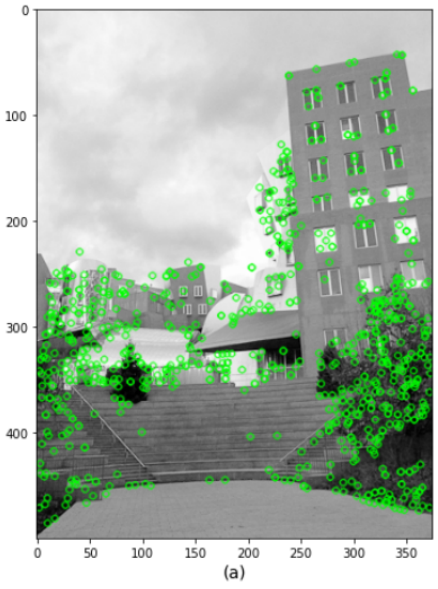

Figure 4.a: Keypoint using SIFT descriptor

One thing to keep in mind is that after extracting the keypoints, you only get information about their position in the image and possibly their coverage area (typically approximated by a circle or ellipse). While keypoint position information can be valuable at times, it does not reveal much about the keypoints themselves.

We will know certain general features of the extracted keypoints whether they are centered around edges, or a blob depending on the technique used to extract keypoints (SIFT, Harris corners, MSER), but you will not know how distinct or similar one keypoint is to the other.

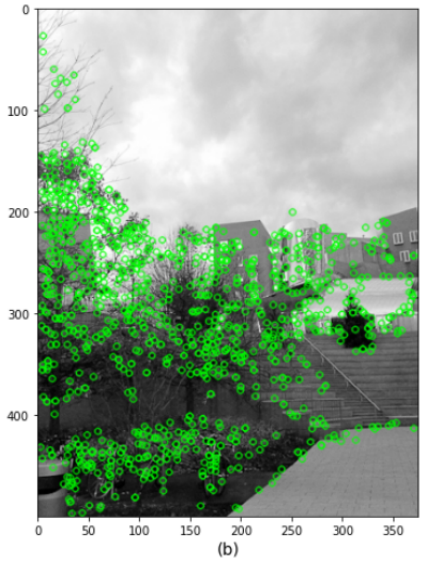

Figure 4.b Keypoint using SIFT descriptor

So, the solution for this is descriptors: they are the means by which the keypoints can be compared. They highlight various keypoint properties in vector format (constant length). It could be their intensity in the direction of their most pronounced orientation, for example. It is giving a numerical description to the area of the image to which the keypoint refers.

We then used FLANN matcher as the result obtained from it was better than compared to BF matcher in terms of speed and approximation of nearest neighbours. A method called the ratio test is suggested by the writers of the SIFT paper to make sure that the features returned by KNN are very similar. We basically go through each of the pairs that KNN gives us and do a distance test for each one. If there is a certain ratio between the distances of features f1 and f2, we keep them. If there isn't, we throw them away. Also, you have to pick the ratio number manually. It's pretty much the same thing that the cross-checking choice in the BruteForce Matcher does. Both checks to see if two detected traits are really close enough to be thought of as the same. Figure 5 shows the BF and FlANN matcher for sample images and from the figure 5.a we can see that the highlighted area in the image are matched incorrectly in BF matcher and in FLANN matcher those points are matched to the top of the building.

5.1. Computing Homography

First, we create keypoints and the feature vector associated with each keypoint in SIFT. Furthermore, if k is a keypoint in image x and k^1 is a keypoint in image x^1, it may be determined whether k and k^1 are excellent matches using the feature vectors. We search $x^1$for a k^1 that is much closer to k than any other keypoint in x^1 in order to find a solid match

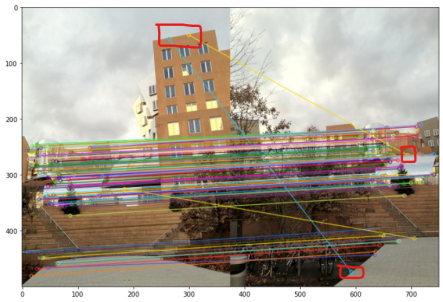

Figure 5.a: BF Matcher

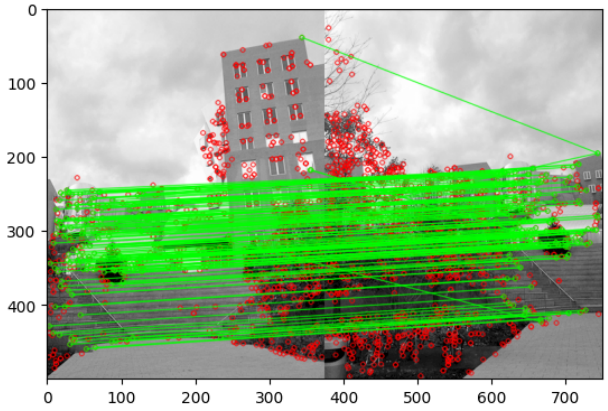

Figure 5.b FLANN Matcher

for each k. A set of potential inliers is formed by the set of good matches.

After that, RANSAC is used. We select four good matches at random, compute a homography from these four, and measure the degree to which the homography is consistent with the number of good matches (i.e., if k,k^1 are good matches, we expect that the homography will map k to something close to k^1 in most situations). For this homography, good matches that are consistent with the homography are called inliers, and those that are not compatible with the homography are called outliers. We determine how many outliers there are in this homography. After that, we repeat the process 1000 times, selecting a fresh set of four good matches each time. At each iteration, we derive a homography and determine how many outliers are connected to it. The homography with the fewest outliers is retained.

The Homography matrix values for the image shown in Figure 4 is as below

```
Homography Matrix for sample image:
 [[ 6.59146646e-01  1.23657522e-01  1.85988387e+02]
 [-3.02167931e-01  9.09941797e-01 -1.62408949e+00]
 [-8.81563130e-04  2.02352109e-06  1.00000000e+00]]
```

We used the estimated homography matrix to warp one image and overlap it onto the other. This process created a seamless panoramic image.

5.2. Results

Figure 6.a Image Pair 1

Figure 6.b Image Pair 2

Figure 6.c Image Pair 3

Figure 6.d Image Pair 4

6. REFERENCES

[1]Brown, M. A. and Lowe, D. (2006). Automatic panoramic image stitching using invariant features. International Journal of Computer Vision, 74(1), 59-73. https://doi.org/10.1007/s11263-006-0002-3

[2] Xiang, T., Xia, G., Bai, X., Zhang, L. (2018). Image stitching by line-guided local warping with global similarity constraint. Pattern Recognition, 83, 481-497. https://doi.org/10.1016/j.patcog.2018.06.013

[3] Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". Alvey Vision Conference. Vol. 15.

[4] Lindeberg, Tony (2015). "Image Matching Using Generalized Scale-Space Interest Points". Journal of Mathematical Imaging and Vision. 52: 3–36. doi:10.1007/s10851-014-0541-0. S2CID 254657377.

[5] https://docs.opencv.org/

[6] Mathematics of stitching. (n.d.). cs.brown.edu. https://cs.brown.edu/courses/csci1950-g/results/proj6/steveg/theory.html