Panorama Image Stitching

Rohit Lal*[1], Khush Agrawal*[1], and Himanshu Patil*[1]

I. Objective

This report has been submitted as a partial requirement towards completing the Digital Image Processing (ECL411) course project. The objective of this project is to study and implement code for Panorama Stitching from scratch. Panorama Stitching is the process of considering multiple images (taken from different viewpoints, with overlapping field of view) and combining (stitching) them, such that the result is a segmented panorama or has a resolution higher than the original images.

Panorama stitching can be essentially divided into four steps: Keypoint detection and feature extraction, descriptor matching, homography estimation, and image wrapping. In the following subsections, we briefly describe these steps and the implementation details we followed. Algorithm 1 systematically described the implementation steps we followed.

II. Theory

Feature detection and matching is a vital task in the field of Computer Vision. A feature is a piece of information about the content of an image. It tells us about the specific properties of the image. Features may be edges, points, objects in the image. An image feature is usually composed of a feature keypoint and a feature descriptor. The keypoint usually contains the patch 2D position and other stuff if available such as scale and orientation of the image feature. The features that are in specific locations of the images, snow mountain peaks, building corners, etc. These kinds of localized features are often called keypoint features (or even corners) and are often described by the appearance of patches of pixels surrounding the point location. These keypoints can be estimated by various methods. Some of them are listed below

- SIFT (Scale-Invariant Feature Transform)
- SURF (Speeded-Up Robust Features)
- FAST (Features from Accelerated Segment Test)
- BRIEF (Binary Robust Independent Elementary Features)
- ORB (Oriented FAST and Rotated BRIEF)
- Harris Corner Detector

We will discuss two most widely used algorithm for feature detection in the below sub-section.

A. *Harris Corner Detector*

Harris Corner Detector is an extremely popular corner detection algorithms used in Computer Vision (CV). This algorithm has garnered immense interest due to its accuracy

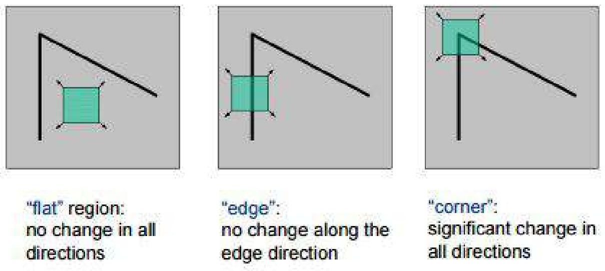
"flat" region:
no change in all directions
"edge":
no change along the edge direction
"corner":
significant change in all directions

Fig. 1. Difference between flat edge and corner. Flat has no change in all directions. Edge has no change along edge direction. Corner has significant change in all directions

and computational efficiency. A corner is a point which has dominant gradient in two or more directions. A corner can be thought as the junction of two edges. These are very essential features in the image which are invariant to rotation, illumination and translation (figure 3) [1].

Let us assume an grayscale image $I(x, y)$ with windowing function $w(x, y)$. $I(x + u, y + v)$ denotes is the intensity at the moved window $(x + u, y + v)$

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

where:

- $I(x, y)$ is the intensity at (x,y)
- $I(x + u, y + v)$ is the intensity at the moved window $(x + u, y + v)$
- $w(x, y)$ is the window at position (x,y)

Since we are looking for windows with corners, we are looking for windows with a large variation in intensity. Hence, we have to maximize the equation above, specifically the term:

$$\sum_{x,y} [I(x+u,y+v) - I(x,y)]^2 \tag{2}$$

Using Taylor series expansion:

$$E(u,v) \approx \sum_{x,y} [I(x,y) + uI_x + vI_y - I(x,y)]^2 \tag{3}$$

Which can be expressed in a matrix form as:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \tag{4}$$

Let's denote:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \tag{5}$$

[1]Department of Electronics and Communication Engineering Visvesvaraya National Institute of Technology, Nagpur, India {khush, rohit, himanshu}@students.vnit.ac.in

*All authors claim equal contribution and the order of appearance of names has been decided on a random basis.

So, our equation now is:.

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{6}$$

A score is calculated for each window, to determine if it can possibly contain a corner:

$$R = det(M) - k(trace(M))^2 \tag{7}$$

where:

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

a window with a score R greater than a certain value is considered a ”corner”

Set a threshold on R and keep all the points above a threshold R. This may cause a lot of point in a small neighbourhood. Hence we find a point of local maxima in the neighbourhood of cluster of points.

B. SIFT feature detection

In section II-A we saw that Harris is rotation-invariant, which means, even if the image is rotated, we can find the same corners. This is so because corners remain corners in the rotated image also. However, these are not scale-invariant. A corner in a small image within a small window can look flat when it is zoomed in the same window. SIFT stands for Scale-Invariant Feature Transform. SIFT is invariant to image scale and rotation. Major advantages of SIFT are

- Distinctive: The features are very distinctive
- Quantity: Even for small objects it can predict many features
- Efficiency: Very fast compute speed
- Locality: Local features are robust to occlusion and clutter

Once the keypoint is detected we try to build descriptor for it. Each keypoint pixel takes account its nearest pixel. This is size of 16×16. This grid is further divided into grid of 4×4. For each of these sub-block 8 bin orientation histogram is created. [2]

C. Feature Matching

The keypoints matching between a pair of two images is performed using the nearest neighbors. However, in a few cases, the first and second closest matches can be extremely nearby, which can be a result of the present noise or some other unknown reasons. In such cases, the ratio of closest to second-closest distances is considered. If this calculated ratio is greater than the threshold value of 0.8, such ambiguities are discarded. It has been statically observed [3] that this procedure eliminates around 90% of false matches, whereas only discarding about 5% correct matches, which in effect improves the overall performance of the algorithm. Feature matching can be done using

- Nearest neighbor techniques
- kd-trees and their variants
- Hashing
- Exhaustive search

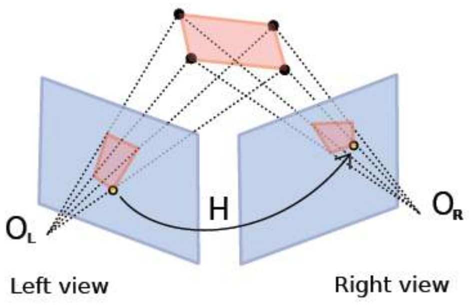

Fig. 2. Homography describes the projective geometry of two cameras and a world plane. In simple terms, homography maps images of points which lie on a world plane from one camera view to another. It is a projective relationship since it depends only on the intersection of planes with lines.

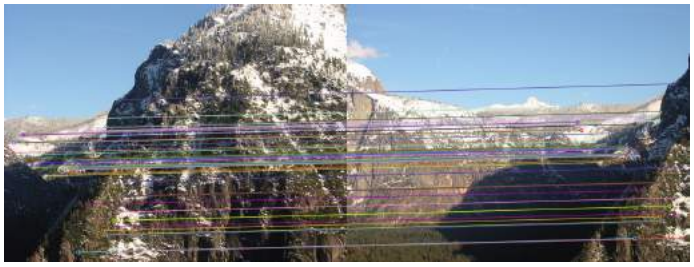

Fig. 3. SIFT feature matching using Brute force method

D. Homography Estimation and Image Blending

Perspective transform is a transform that helps in mapping any two projection planes with the same center of projection called Homography. This is Represented as $3x3$ matrix in homogeneous coordinates [4].

E. Homography Estimation and Image Blending

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} \tilde{x}_d \\ \tilde{y}_d \\ \tilde{z}_d \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \tag{8}$$

For a given pair i of corresponding points:

$$x_d^{(i)} = \frac{\tilde{x}_d^{(i)}}{\tilde{z}_d^{(i)}} = \frac{h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}} \tag{9}$$

$$y_d^{(i)} = \frac{\tilde{y}_d^{(i)}}{\tilde{z}_d^{(i)}} = \frac{h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{123}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}} \tag{10}$$

$$A = \begin{bmatrix} x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \end{bmatrix}$$

$$h = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{bmatrix}^T$$

Solve for h: $Ah = 0$, such that $||h|| = 1$ Define least squares problem:

$$\min_h ||Ah||^2 \tag{11}$$

such that $||h|| = 1$. We know that:

$$||Ah||^2 = (Ah)^T(Ah) \;\;\&\;\; ||h||^2 = h^T h = 1 \tag{12}$$

$$\min_h (h^T A^T Ah) \; such \; that \; h^T h = 1 \tag{13}$$

Define Loss function $L(h, \lambda)$:

$$L(h, \lambda) = h^T A^T Ah - \lambda(h^T h - 1) \tag{14}$$

Taking derivatives of $L(h, \lambda)$ w.r.t h: $2A^T Ah - 2\lambda h = 0$

$$A^T Ah = \lambda h \tag{15}$$

Eigenvector h with smallest eigenvalue λ of matrix $A^T A$ minimizes the loss function $L(h)$.

It may be possible that some of the images may have wrong matches. So we need to filter our such wrong matches. To do this, we use Random Sample Consensus (RANSAC) to separate points of inlier and outliers. These are the following key steps of the RANSAC algorithm. We first select four pairs of matching points from N pairs randomly to establish equations. Then we find the solution of M's unknown parameters. Hence, calculate the distance between the two points of N-4 pairs. If the computed distance is smaller than a set threshold, the two selected points can be labeled as the interior points. Further, consider four different pairs, and repeat the aforementioned process. Finally, the optimal solution will be the parameter that has the maximum number of interior points. The details of RANSAC can be found in [5].

Once we get the homography matrix, we start placing the image on the canvas one by one, starting from left to right.

Algorithm 1:

```
Read given images into an array.
Convert all images into gray-scale.
Order the image array.
Create a canvas to accommodate the panorama.
Add the first image of the array to the canvas.
Indexer = 0.
while all images not processed do
    Consider the canvas image and find
      correspondences with image in the array at
      Indexer.
    Find the homography (H) using RANSAC
    H ⟵ H × T .
    project the first image on the canvas.
    project the translated image on the canvas.
end
Find contour of large panorama image
Fit minimum area rectangle
Crop image to minimize black area
```

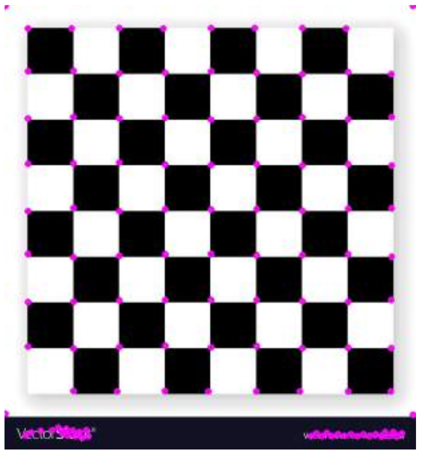

Fig. 4. Harris Corner points detected in Chess image

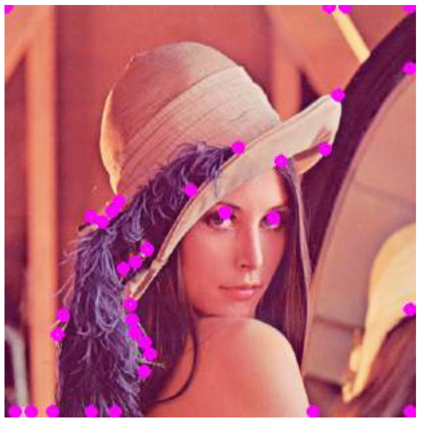

Fig. 5. Harris Corner points detected in Lenna image

III. Results and Discussion

A code for Harris corner detection was written from scratch and its results is shown in figure 4 and figure 5 [1]. Figure 6 presents a qualitative comparison between panoramic images obtained from the SIFT keypoint matching and Harris keypoint matching. It can be observed that using Harris keypoint gave poorer results compared to SIFT features. This discrepancy in the results is because SIFT is better at handling images at different resolutions due to its scale-invariant property. For Harris feature matching, we used Histogram of Oriented Gradients (HoG) descriptors obtained from the OpenCV library's inbuilt functions [6]. Since we added an extra step to order the input images, our method is invariant to the order of input images and can stitch the images together in the same manner regardless of the input order.

Currently, the implemented code's applicability is limited to the case of a flat panorama. For extending it to a cylindrical panorama, additional steps are required like camera calibration, i.e., finding intrinsic and extrinsic camera parameters. The stitched image has noticeable seams in between. To avoid this issue, we may need to use blending methods like alpha blending.

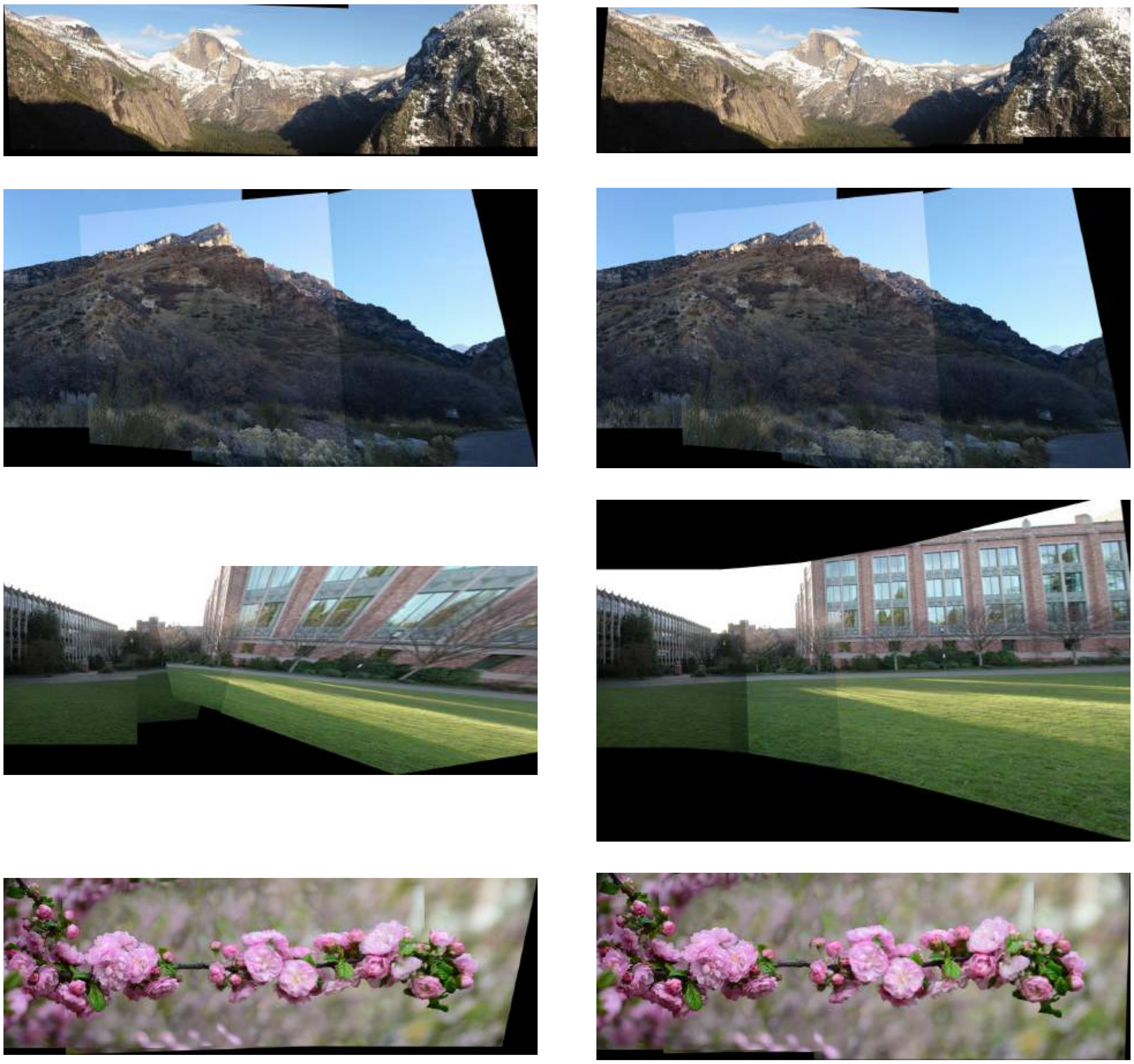

Fig. 6. The images show various panorama stitching of images. The left image shows stitching using Harris keypoints. Whereas the right image shows stitching using SIFT keypoints. Since SIFT is scale invariant, it produces better results compared to Harris keypoints

References

[1] C. Harris and M. Stephens, “A combined corner and edge detector,” in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[2] “Introduction to sift (scale invariant feature transform),” https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40, note = Accessed: 2010-09-30.

[3] “OpenCV official documentation,” https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html, accessed: 2010-09-30.

[4] “Image features, homographies, ransac and panoramas, tali dekel, csail, mit,” http://6.869.csail.mit.edu/fa17/lecture/lecture14sift_homography.pdf, note = Accessed: 2010-09-30.

[5] Jubiao Li and Junping Du, “Study on panoramic image stitching algorithm,” in *2010 Second Pacific-Asia Conference on Circuits, Communications and System*, vol. 1, 2010, pp. 417–420.

[6] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, 2005, pp. 886–893 vol. 1.