Reinelle Jan Bugnot
MSAI,
Matric No: G2304329L
email: bugn0001@e.ntu.edu.sg

Rohin Kumar Maheswaran
MSAI,
Matric No: G2303513K
email: rohinkum001@e.ntu.edu.sg

Aradhya Dhruv
MSAI,
Matric No: G2303518F
email: ar0001uv@e.ntu.edu.sg

Image Stitching Using Homography

Image stitching, also known as photo stitching, is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image. This process is commonly performed through the use of computer software, and most approaches to image stitching require nearly exact overlaps between images and identical exposures to produce seamless results. In this report, we discussed, in comprehensive detail, the steps of performing image stitching given two images using a widely known algorithm called Scaled-Invariant Feature Transform (SIFT).

Keywords: Image Stiching, SIFT, Difference of Gaussians, Feature Detection, Feature Matching, Homography

1 Introduction

In the field of computer vision, image stitching is the process of combining multiple, often overlapping, images to create a single, larger, and more comprehensive image or panorama. This technique is widely used to extend the field of view beyond what can be captured in a single photograph. It is a fundamental concept in computer vision and image processing with numerous practical applications, such as autonomous robotics, medical imaging, virtual reality, and photography. In order to highlight the importance of image stitching in providing a more engaging and educational visual experience, we will examine the fundamentals and applications of image stitching in computer vision in this report [1].

2 Feature Detection

In order to overlap two images of the same scene captured in two different view points, we need to perform 3 processing steps; namely, Feature Detection, Feature Matching, and Image Stitching. Each step plays a crucial role in ensuring a successful and visually pleasing result.

2.1 Feature Detection using SIFT. Feature Detection is the process of identifying distinctive points or regions in the images being analyzed. These features are essential for establishing correspondences between the images and aligning them accurately. In this report, we implemented the Scale-Invariant Feature Transform (SIFT) algorithm to detect a set of unique feature points from two input images used for the image stitching process. In order to have a comprehensive understanding and appreciation of how SIFT operates in the background, we decided to perform SIFT Feature Detection manually through the use of low-level computer vision libraries.

The primary logic behind feature detection is the dea that real world objects are visually recognizable only at a certain scale. For instance, you might see an ant on the table if you look really closely, but if you're standing a few meters away, then the same ant might not be visible at all. SIFT tries to mimic this multi-scale nature of vision on digital images via a representation called *scale-space* [2]. The scale space of a given image is defined by a Laplacian function $L(x, y, \sigma)$ that is generated through the convolution of a Gaussian kernel, as shown in Equation 1 [1]. In python, this can easily be implemented using the $cv2.GaussianBur()$ function of OpenCV.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

This equation is applied at different Gaussian intervals with input images from different octaves (typically, 4 octaves). In effect, each image within an octave is progressively blurred with increasing intensity (as defined by the σ parameter). The octave of images are then scaled down progressively, where each subsequent octave is half the size of the previous one. The result of this step is a so-called *image pyramid*, as shown in Fig. 1.

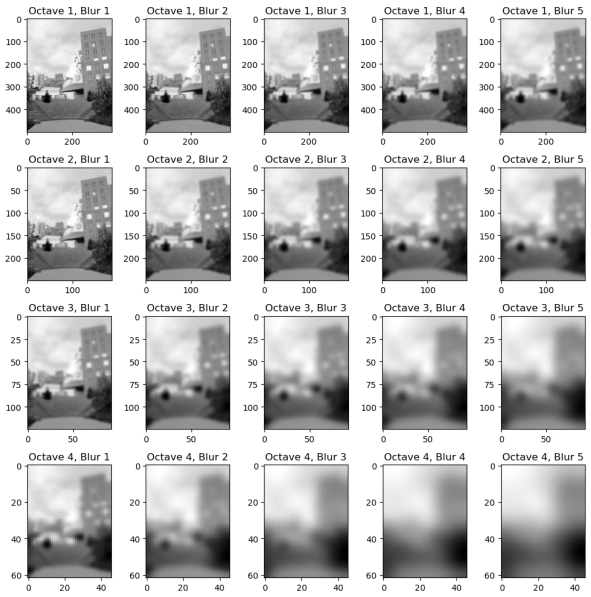

Fig. 1 Image Pyramid representing the scale space.

The next step is to detect stable keypoint locations in the image using the generated scale space. The SIFT algorithm implements a technique called the *Difference of Gaussians*, $D(x, y, \sigma)$, by computing the difference between two consecutive images within the same octave in our image pyramid [1]. More specifically,

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2}$$

The output of this process is shown in Fig. 2. The Difference of Gaussian (DoG) is used to approximate a scale-normalized Laplacian of Gaussian which yields the SIFT algorithm's scale-invariant property [1].

From the DoG images, we need to calculate the scale-space extrema which will be our candidate feature points for the entire

Version 1.18, November 1, 2023

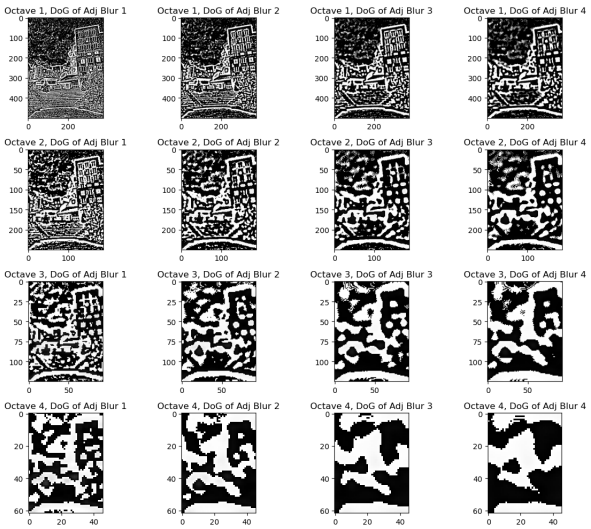

Fig. 2 Difference of Gaussian Output with Different Values of Adjacent Blur

image. To do this, each pixel in each DoG image in each octave is compared to its 8 neighbors as well as the 9 corresponding pixels of the previous and the next image in the octave of DoG images, as illustrated in Fig. 3

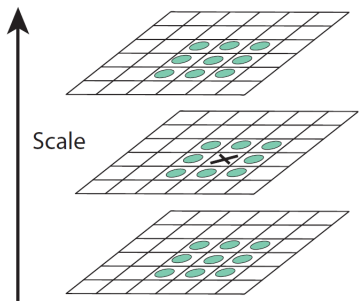

Fig. 3 Point of interest for Extrema Detection [3]

This is done by extracting (at most) a 3x3x3 cube of pixels around a point of interest in the DoG image of a given octave. This point of interest (in the center of the 3x3x3 cube) will be considered an "extrema" if and only if it is the largest or smallest value in this cube [3]. This logic is applied across all DoG images, across all octaves. We recorded all the positions that were considered extrema points as our candidate feature points for the original image. It is important to note, however, that since the scale-space is comprised of multiple octaves of varying scales, the identified extrema point pixel locations must then be rescaled back to the scale of the original image (by reversing the downscaling logic applied during the generation of the image pyramid). A simplified pseudocode that we used to perform this task, as implemented in the original SIFT paper [3], is shown in algorithm 1, while Fig. 4 shows an example of the identified extrema points for one DoG image, and its corresponding rescaled counterpart in the original image.

After recording all of the identified keypoints across all DoG images across all octaves, we are left with a very large collection of keypoints that overpopulates the original image, providing very little meaningful information, as shown in Fig. 5. Different strategies can be implemented to filter out unwanted keypoints. Particularly, in the original SIFT paper, keypoints that lie on low contrast areas and edges were removed. This was done by using the derivative

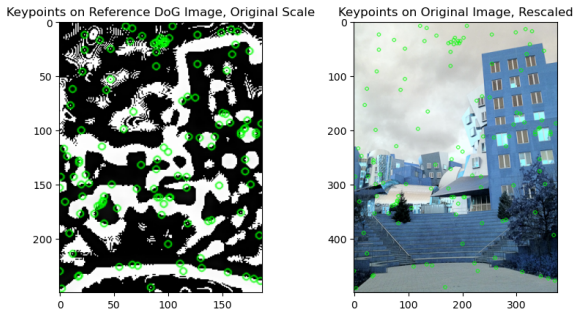

Fig. 4 Identified Keypoints for one DoG Image Sample

Algorithm 1 Detect Keypoints from DoG Images

```
keypoints ← []
for octave in DoG do
    for DoG_image in octave do
        for each image_row in DoG_image do
            for each pixel in image_row do
                Get pixel coordinate
                Get region of Interest
                if pixel is extrema in region of interest then
                    rescale coodinate values to match image
                    keypoints ← rescaled pixel coordinate
                end if
            end for
        end for
    end for
end for
```

and Hessian of the Taylor expansion of the scale-space function, $D(x, y, \sigma)$ to estimate the function value at the extremum, $D(\hat{x})$, given by Equation 3 [3].

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{3}$$

From here, the authors of the SIFT paper are able to filter out low contrast points by rejecting any keypoint with a corresponding $|D\hat{x}|$ value less than 0.03. Afterwards, the author filtered out keypoints corresponding to edges by thresholding a constant parameter that defines the ratio between the Trace and Determinant of the Hessian of $D(x, y, \sigma)$, which is a rather complicated and lengthy mathematical process to implement manually. Hence, for the purpose of simplicity, we decided to use the $cv.cornerHarris()$ module of OpenCV to calculate the corner response R of a given pixel in the image [4].

$$R = det(M) - k(trace(M))^2 \tag{4}$$

A very small magnitude of R from Eq. 4 indicates that the observed keypoint lies on a *flat* or low contrast area, while a negative value signifies that the keypoint lies on an edge [4]. Therefore, we can set a relatively low threshold value, and reject any keypoint whose corner response R is less than our specified threshold. Doing this filters majority of the unwanted keypoints generated in the previous step, resulting in keypoints that better represent important features of the original image as shown in Fig. 6, and completing the Feature Detection step for Image Stitching.

The OpenCV library has module for SIFT that packages all of the steps detailed above into only a few lines of code. Comparing the results of our manually implemented feature detection algorithm and the functions provided by the module, as shown in Fig. 7, we can clearly see that our implementation is well at par with OpenCV. However, for the purpose of computational and coding

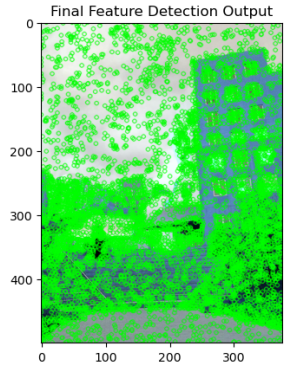

Fig. 5 All Identified Keypoints (Rescaled)

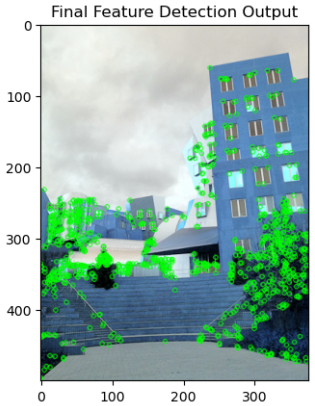

Fig. 6 All Identified Keypoints (Rescaled and Filtered)

efficiency in the succeeding steps of the image stitching process, we decided to utilize the OpenCV library for feature detection moving forward.

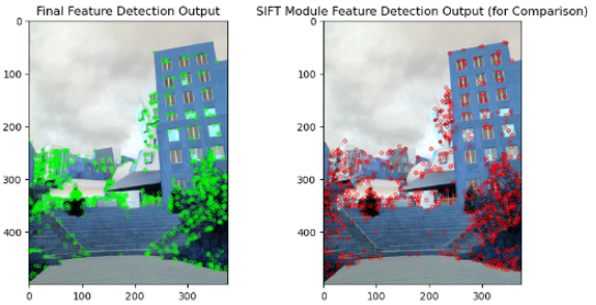

Fig. 7 Hardcoded Feature Detection VS Feature Detection through SIFT Module

3 Feature Description and Matching

Once we have detected key feature points in our images, the next crucial step is to describe these features and find corresponding matches between images. This process ensures accurate alignment during image stitching.

3.1 Best Match using Brute Force. In order to identify the best matches, brute force matching compares each keypoint in one image with every keypoint in another image. A distance metric between the feature descriptors is frequently used as the basis for the matching criterion. These descriptors are typically shown as high-dimensional vectors in the SIFT context.

The best match is determined by the algorithm, which measures the distance between each keypoint's descriptor in the two photos. Although finding the most comparable keypoints is guaranteed by this method, it can be computationally expensive, particularly when dealing with a large number of keypoints.

3.2 Key Points Matching using KNN (K-Nearest Neighbors). The K-Nearest Neighbors (KNN) technique is frequently used to increase efficiency. KNN determines the top k matches for each keypoint based on descriptor distances rather than identifying the one best match.

The number of nearest neighbors to take into account is denoted by "k" in this context. The theory is that there is a good chance that several keypoints in the second image will match well if their descriptions are close to those of a keypoint in the first image.

The optimal matches are chosen by a filtering procedure that is conducted after determining the k-nearest neighbors for every keypoint. This may entail comparing the first- and second-nearest neighbor distance ratios and eliminating matches with ratios higher than a predetermined threshold.

4 Homography and Image Stitching

4.1 Homography Matrix. The role of the homography matrix in image stitching is to estimate the transformation between two images taken from different viewpoints. The homography matrix is a 3x3 matrix that describes the transformation between two planes in a 3D space. It can be interpreted as rotation along x, y, z direction and translation along x and y directions, keeping z-direction translation constant to 1. In image stitching, the homography matrix is used to align two images so that they can be merged into a single panoramic image. One the matrix is computed, one of the images is transformed to match the perspective of the other image using **perspective warping**. The transformed image is then blended with the other image to create a seamless panoramic image. Since Homography is a 3X3 image, we can write it as:

$$\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

Let us consider the first set of corresponding points (x_1, y_1) in the first image and (x_2, y_2) in the second image. Then, the Homography H maps them in the following way:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

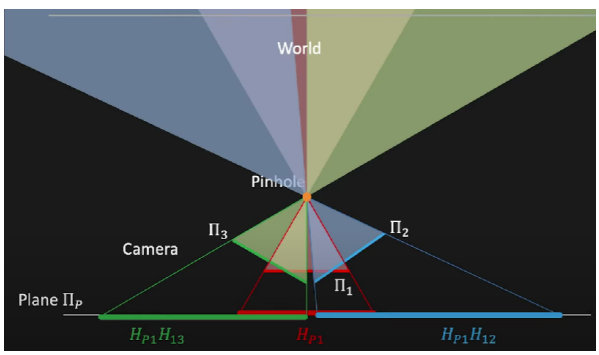

Fig. 8 Pictorial Representation of Homography Composition

To understand homography graphically refer to Figure 8. Consider that the planes $\pi 1$, $\pi 2$ and $\pi 3$ are three different angles through which the overlapping images are taken. If we project the three planes backwards on the plane π' then we can get a panoramic view of the individual images captured by planes $\pi 1$, $\pi 2$ and $\pi 3$

4.2 Image Stitching. It is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image. The purpose of image stitching is to solve the field of view limitation of images and videos and create a single image that includes a larger field of view. Image stitching is used to create a seamless panoramic image by aligning and blending multiple images together. The process of image stitching involves three main components: image registration, calibration, and blending. Image stitching is widely used in modern applications such as document mosaicing, image stabilization feature in camcorders, high-resolution photomosaics in digital maps and satellite imagery, medical imaging, multiple-image super-resolution imaging, video stitching, and object insertion.

4.3 Blending Techniques and Perspective Warping. Image blending is an essential step in the image stitching process to seamlessly merge overlapping images into a cohesive panorama.

- **Weighted Averaging:** In this method, each pixel in the overlapping region is assigned a weight, and the final pixel value is computed as a weighted average of the corresponding pixel values in the input images. Weights can be determined based on factors like distance from image borders or proximity to keypoints.
- **Feathering:** Feathering involves smoothly transitioning pixel values from one image to another in the overlapping region. A gradient mask is applied, gradually reducing the influence of one image while increasing the influence of the other. This creates a natural-looking blend.

5 End Result:

Please refer to Figures [9, 10, 11 and 12]

Fig. 9 Stitched Image 1

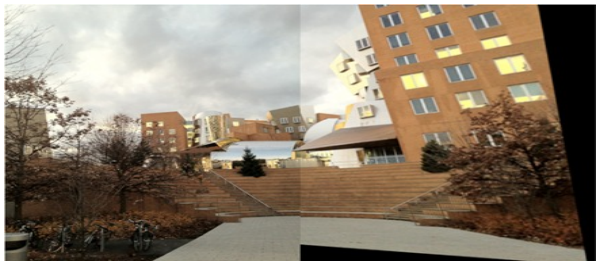

Fig. 10 Stitched Image 2

Fig. 11 Stitched Image 3

Fig. 12 Stitched Image 4

6 References

1 Lu Shijian. AI6121 Computer Vision - Lecture 5: Point Detection. 2023
2 Deepanshu Tyagi. Introduction to SIFT(scale invariant feature transform). Apr. 2020. url: Link
3 David G Lowe. "Distinctive image features from scale-invariant keypoints". In: International journal of computer vision 60 (2004), pp. 91–110.
4 Harris Corner Detection OpenCV Documentation. url: link
5 OpenAI. (2020). ChatGPT - An AI-Powered Language Model. Retrieved from url: Link

List of Figures

List of Tables