IMAGE STITCHING

Jamwal Snigdha, Ramanathan Thanya

Matric. No.: G2304207H, G2303746K
Email: SNIGDHA001@e.ntu.edu.sg, THANYA001@e.ntu.edu.sg

1 Introduction

Image stitching is a method where several overlapping images are combined to produce a single, seamless panoramic or wide-angle image. In order to increase the field of vision beyond what can be captured in a single frame, this technique is frequently employed in photography, mapping, virtual tours, and many other applications. The basic processes in picture stitching include blending, geometric transformation estimates, and feature matching and detection.

Key points or interest points are found in each input picture and their descriptors are derived during the feature detection and matching process. The appropriate spots in the overlapping regions are then found by matching these key points across photos. To correctly align the pictures, geometric transformations including translation, rotation, and scaling are predicted once matching points have been determined. The purpose of these modifications is to precisely integrate the pictures by warping or transforming them. Lastly, blending techniques are used to eliminate seams and provide a panorama with a unified aesthetic. To accomplish picture stitching, a variety of algorithms and software tools are available; they vary from simple approaches to more complicated ones that can handle complex instances and adjust for lens aberrations.

1.1 Importance and Applications

Image stitching plays a crucial role in various fields due to its significance in creating seamless panoramic and wide-angle images. The use of photography and videography is one of the main applications. It makes it possible for filmmakers and photographers to capture amazing interiors, exteriors, and architectural details that are impossible to convey in a single shot. This enables for more thorough and engaging narrative in addition to improving the content's visual attractiveness. Apart from photography, image stitching is widely employed in satellite imaging and aerial photography to generate intricate, high-definition maps and pictures covering vast regions. It helps with jobs like environmental monitoring, urban planning, and catastrophe management, where precise and broad-field perspectives are necessary for analysis and decision-making.

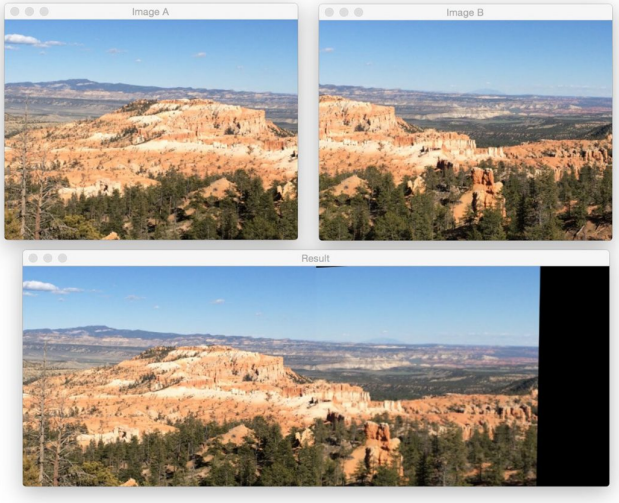

Fig. 1. Example Output of an Image Stitching Algorithm

Furthermore, robotics and computer vision have benefited from the use of image stitching. It is used to give robots, drones, and autonomous cars panoramic vision so they can see their surroundings more clearly. This is necessary for object identification, obstacle avoidance, and navigation. Image stitching is used in radiology to provide panoramic views of scanned pictures, which makes it easier to analyse organs and tissues thoroughly in the medical profession. To put it simply, picture stitching improves our capacity to gather, process, and decipher visual data across a range of applications, which helps us make better decisions and solve problems.

1.2 Prior Research

Over the years, there have been major advancements in previous image stitching research, with an emphasis on increasing the process's accuracy and efficiency. Early research focused mostly on fundamental methods of picture alignment and blending. With these systems, control points had to be manually selected and users had to find related characteristics in overlapping pictures. However, human assistance was necessary and this process took a long time. Automating feature matching and detection was the focus of later research,

which produced algorithms like Speeded-Up Robust Features (SURF) [3] and Scale-Invariant Feature Transform (SIFT). These methods greatly increased picture alignment's practicality and user-friendliness by improving its accuracy and dependability.

Subsequent developments in the field of image stitching have primarily addressed difficult situations, like managing image sequences captured by moving cameras or correcting lens distortions. Researchers have investigated sophisticated algorithms, such as bundle adjustment and homography estimation, to lessen distortions and geometric transformations. The ability to produce increasingly intricate and immersive panoramas was expanded with the introduction of multi-row and multi-level image stitching techniques. Additionally, new opportunities for creating both 3D representations of scenes and wide-angle panoramas have been made possible by the integration of image stitching with 3D reconstruction and depth estimation techniques.

Deep learning methods have been used in recent studies to improve image stitching. Convolutional neural networks (CNNs) have been utilised for various tasks like seam estimation and feature matching, which have resulted in enhanced stitching quality and resilience to changes in lighting, scene complexity, and viewpoint. Additionally, machine learning algorithms have been used to improve image blending, minimise visible seams, and enhance the stitched panoramas' overall visual quality.

2 Methodology and Implementation

Image stitching is essentially a geometric transformation problem, where the goal is to find the correct transformation that aligns and blends multiple images together to create a seamless panorama. The process can be represented as follows:

Feature Detection and Matching The first step is to identify the salient characteristics or points of interest in each image given a set of images. These characteristics can be expressed as two-dimensional (x, y) coordinate points. Finding correspondences between these features in overlapping areas of neighbouring images is the task that follows feature detection. This can be mathematically expressed as matching points between images, that is, locating pairs of points that correspond to the same feature in two different images: (x2, y2) and (x1, y1) in one.

Geometric Transformation The next step is to compute the geometric transformation that projects one image onto another after feature correspondences have been determined. Translation, rotation, scaling, and homography (perspective transformation) are examples of common transformations. These conversions can be expressed mathematically as matrices or equations that map the point coordinates from one image to the other.

Warping and Blending Once the transformation has been determined, the images are appropriately aligned by warping or transforming them. In order to map each pixel in the source image to the destination image, the calculated transformation must be applied to each pixel. To achieve a seamless transition between the images, they must first be aligned. Methods such as gradient-based blending or weighted averaging may be used for this.

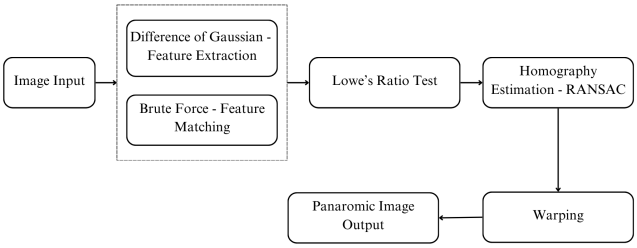

Fig. 2. Process Flow of Algorithm

2.1 Difference of Gaussian

The Difference of Gaussian (DoG) is a key technique used in image stitching, specifically in the context of feature detection. It is a technique for emphasising and locating unique local characteristics or points of interest in pictures. Finding and matching these features between several images is essential to aligning and blending them together seamlessly during the image stitching process.The DoG is computed by subtracting the two blurred images

To highlight areas with different textures or intensities, the DoG computes the difference between two Gaussian-blurred versions of the image. Finding probable feature points is made simpler by these variations, which highlight edges and other important picture structures. The first step in using DoG for image stitching is to create image pyramids by down sampling the original image into several scales. The DoG is calculated for each scale by deducting one Gaussian-smoothed version from another. Local maxima in these images frequently correspond to potential feature points, and these DoG images function as a multi-scale representation of the input image.

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

The DoG is computed by subtracting the two blurred images:

$$\text{DoG}(x, y; \sigma_1, \sigma_2) = G(x, y; \sigma_1) - G(x, y; \sigma_2)$$

When it comes to image stitching, the DoG-detected features assist in locating corresponding points across various images. Once these matches are found, the images can be correctly aligned by computing geometric transformations. The use of DoG in feature detection is essential for enhancing the stitching process's robustness and accuracy, which makes it possible to produce seamless panoramic images.

2.2 SIFT Feature Extractor

The Scale-Invariant Feature Transform (SIFT) is a highly influential feature extraction technique in computer vision. Developed by David Lowe in 1999, SIFT [2] is designed to identify and describe distinctive features or keypoints in images, which are invariant to changes in scale, orientation, and partial occlusion. SIFT is widely used in various applications, including object recognition, image matching, and image stitching.

The process of SIFT feature extraction involves several crucial steps. The image is first converted into a multi-scale representation known as a Gaussian pyramid by repeatedly applying Gaussian blurring and downsampling. SIFT can handle differences in the size and scale of features thanks to this pyramid. The Difference of Gaussians (DoG) is then calculated at each scale level by deducting two neighbouring Gaussian pyramid scales. Potential keypoints can be identified by the DoG images, which highlight areas with notable variations in pixel intensity.

SIFT extracts keypoint descriptors from the DoG images, which are then utilised for matching and recognition. A local picture patch is taken into consideration for every keypoint, and gradients and orientations are calculated inside this patch. The keypoint's local neighbourhood is characterised by a unique descriptor that is generated based on the gradient magnitudes and orientations, rendering it resistant to modifications in scale and rotation.

```
def detectAndDescribe(image):
  # convert the image to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  # detect and extract features from the image
  descriptor = cv2.xfeatures2d.SIFT_create()
  (kps, features) = descriptor.detectAndCompute(image, None)
  kps = np.float32([kp.pt for kp in kps])
  # return a tuple of keypoints and features
  return (kps, features)
```

Fig. 3. Implementation of SIFT and DoG for feature detection and description

2.3 Brute Force Matching

Brute Force matching is a straightforward and intuitive method for comparing and matching descriptors, such as those extracted using feature detection techniques like SIFT or SURF, in computer vision and image processing. It involves exhaustively comparing each descriptor in one set to every descriptor in another set to find the best matches. This approach is simple and conceptually easy to understand, but it can be computationally expensive, especially for large datasets.

A similarity metric—in this case, the Euclidean distance—is computed between each descriptor in the first set and each descriptor in the second set in the Brute Force matching process. Matches are defined as descriptor pairs with the smallest distance. Brute Force matching's primary benefits are its adaptability and simplicity; it can be used with a variety of feature detection algorithms and applied to any kind of descriptor.

Euclidean Distance, $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

2.4 Lowe's Ratio Test

In computer vision applications, Lowe's ratio test is a post-processing technique that is frequently used in feature matching to remove false positive matches between keypoints, especially in the context SIFT and related algorithms. David Lowe introduced it to increase keypoint matching's dependability.

The ratio test's basic concept is to evaluate a possible match's quality by contrasting it with the second-best match's quality. A match is deemed reliable if the ratio of the distance between the best and second-best matches' descriptor vectors is higher than a predetermined threshold. On the other hand, the match is probably ambiguous and is therefore regarded as a false positive if this ratio is below the threshold.

By requiring a certain level of uniqueness in the matches, the ratio test efficiently aids in the removal of false positives in feature matching. Due to scene ambiguities or feature descriptor limitations, multiple keypoints from one image may find similar matches in another image, leading to false positives. Lowe's ratio test improves the overall accuracy and reliability of feature matching, which is crucial in tasks like object recognition, image alignment, and 3D reconstruction. It does this by setting an appropriate threshold, usually around 0.8, which guarantees that only matches with a significant distinction between the best and second-best candidates are retained. The ratio chosen for the algorithm built in this project is 0.75.

2.5 Homography using RANSAC

Homography estimation using the Random Sample Consensus (RANSAC) algorithm is a fundamental technique in computer vision and image processing for finding the perspective transformation that relates two images taken from different viewpoints. The process is often used in tasks like image stitching, object recognition, and augmented reality. RANSAC helps robustly estimate the homography matrix by iteratively selecting a minimal set of correspondences and

identifying the model that best fits these correspondences while ignoring outliers.

Initially, a minimal subset of correspondences is chosen at random by the RANSAC algorithm; in the case of a homography matrix (2D to 2D transformation), this minimal subset is usually four. On the basis of this subset, the homography matrix is then computed. The degree to which the model fits the remaining correspondences is used to evaluate the model's quality. Outliers are defined as correspondences that fall outside of a predetermined threshold of the transformed points, while correspondences inside this threshold are regarded as inliers. The model with the greatest number of inliers is regarded as the best estimate after the procedure is repeated a predetermined number of times.

```
def featureMatching(kps1, kps2, features1, features2, ratio, reprojThresh):
  # compute the raw matches and initialize the list of actual
  # matches
  matcher = cv2.DescriptorMatcher_create("BruteForce")
  rawMatches = matcher.knnMatch(features1, features2, 2)
  matches = []
  # loop over the raw matches
  for m in rawMatches:
    # ensure the distance is within a certain ratio of each
    # other (i.e. Lowe's ratio test)
    if len(m) == 2 and m[0].distance < m[1].distance * ratio:
      matches.append((m[0].trainIdx, m[0].queryIdx))
  # computing a homography requires at least 4 matches
  if len(matches) > 4:
    # construct the two sets of points
    pts1 = np.float32([kps1[i] for (_, i) in matches])
    pts2 = np.float32([kps2[i] for (i, _) in matches])
    # compute the homography between the two sets of points
    (H, status) = cv2.findHomography(pts1, pts2, cv2.RANSAC,
      reprojThresh)
    # return the matches along with the homograpy matrix
    # and status of each matched point
    return (matches, H, status)
  # otherwise, no homograpy could be computed
  return None
```

Fig. 4. Implementation of Brute Force and Lowe's Ratio Test for Feature Matching and Homography

```
def stitchImages(img1Path, img2Path, ratio=0.75, reprojThresh=4.0, direction = "horizontal", blending = False):

  img1 = cv2.imread(img1Path)
  img2 = cv2.imread(img2Path)
  #detect keypoints and extract local invariant descriptors from them
  (kps1, features1) = detectAndDescribe(img1)
  (kps2, features2) = detectAndDescribe(img2)
  # match features between the two images
  M = featureMatching(kps1, kps2,
    features1, features2, ratio, reprojThresh)
  # if the match is None, then there aren't enough matched keypoints to create a panorama
  if M is None:
    return None
  # otherwise, apply a perspective warp to stitch the images together
  (matches, H, status) = M
  if direction == "horizontal":
    result = cv2.warpPerspective(img1, H,(img1.shape[1] + img2.shape[1], max(img1.shape[0], img2.shape[0])))
  elif direction == "vertical":
    result = cv2.warpPerspective(img1, H,(img1.shape[0] + img2.shape[0], max(img1.shape[1], img2.shape[1])))
  result[0:img2.shape[0], 0:img2.shape[1]] = img2

  # return the stitched image
  return result
```

Fig. 5. Image Stitching Function

3 Output Study

The different steps are put together and tested on the given image samples.

Fig. 6. Output of Image Pair 1

Figure 6 is the output obtained for the first given pair. The algorithm has performed considerably well for this image pair and the output panorama has well adjusted features.

Fig. 7. Output of Image Pair 2

Fig. 8. Output of Image Pair 3

Figure 7 and 8 are the outputs obtained for the second and third given pair respectively. While the stitching algorithm has worked well on these pairs as well, there is a noticeable difference in the lighting in the images. This makes the image look unnatural. Hence, the exposure should be altered before carrying out the stitching algorithm.

Fig. 9. Output of Image Pair 4

Figure 9 is the output obtained for the fourth pair. The algorithm has performed considerably well for this image pair as well since both the images in the pair have similar tones and brightness.

4 Inferences

From the images and their respective outputs, it is notable that the Image Stitching algorithms have performed considerably well in fulfilling their objective, However, there are some aspects inferred from the outputs that require attention.

Color and Exposure Variations Variations in color balance and exposure between images are leading to noticeable seams in the final panorama. Image enhancement and color correction is required.

Parallax and Perspective Shifts When capturing images from different viewpoints or angles, parallax and perspective shifts can occur. These shifts can cause distortions and misalignment in the stitched image. This is noticeable in the output image if image pair 2.

Computational Cost The parts chosen can be computationally expensive. Hence, more computationally efficient methods should be chosen

5 Improvements

Using the Features Invariant technique studied in [1] in OpenCV to stitch the images together. Figure 10 displays a clear difference as compared to the original algorithm. This method handles the distortion. The exposure in the image is altered as well so that colors across the image are balanced. Additionally, the improved algorithm doesn't require a direction input, i.e., the images can be input in any order and the algorithm presents the stitched image. The improved algorithm, performs substantially better as compared to the original algorithm.

Fig. 10. Output of Image Pair 2 after improvements

6 References

[1] Brown, M., Lowe, D.G. Automatic Panoramic Image Stitching using Invariant Features. Int J Comput Vision 74, 59–73 (2007). https://doi.org/10.1007/s11263-006-0002-3

[2] 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

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, Speeded-Up Robust Features (SURF), Computer Vision and Image Understanding, Volume 110, Issue 3, 2008, Pages 346-359, ISSN 1077-3142, https://doi.org/10.1016/j.cviu.2007.09.014.