Feature-Based Image Stitching

Mikail Deniz Cayoglu (3437201, Master in Data and Computer Science),
Ha Van Dang (3699676, Master in Scientific Computing),
Sima Esmaeili (3766857, Master in Data and Computer Science)

Abstract

Image stitching is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image. In this paper, we explore different feature-based image stitching algorithms. We explain the fundamental concepts behind feature-based image stitching, including feature detection, image matching, and image alignment. We evaluate the performance of these algorithms, analyse their accuracy, processing time and use cases. The goal is to provide a comprehensive overview of feature-based image-stitching algorithms.

I. Introduction

IMAGE stitching is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image. [14] This can be done using one of the two main approaches. The first approach is *direct*, the second is *feature-based*. In direct image stitching, the algorithm uses all available image data to estimate the camera parameters and stitch the images together. However, direct methods require a close initialization and are sensitive to errors and noise. In contrast, feature-based image stitching uses common features across multiple images to align them and create a seamless panorama. The latter method had the advantage of being more robust against scene movement. [6] In feature-based image stitching, the first step is to detect and extract features that are invariant to changes in lighting, viewpoint, and scale. These features can be detected using various techniques, such as *Harris corner* detection, *SIFT* or *ORB*. Once the features are detected, they are matched across the images using algorithms such as *RANSAC* [2] to estimate the camera parameters. Finally, the images are blended together to form a seamless panorama.

In recent years, many researchers have proposed different approaches for image stitching, including both direct and feature-based methods. For example, some researchers have explored deep-learning techniques for feature-based image stitching [9], [10] and [3]. Convolutional neural networks (*CNNs*) have been used to learn descriptive features that are more robust to variations in lighting and viewpoint, and generative adversarial networks (*GANs*) have been used to generate high-quality panoramic images from a set of input images. Additionally, researchers have developed methods for handling large image sets with varying degrees of overlap, such as those found in surveillance and monitoring systems. These methods aim to select the best subset of images to use for stitching or partitioning the images into smaller groups that can be stitched independently. In our paper, we implemented algorithmic techniques for feature-based image stitching, including *Harris*, *SIFT*, *KAZE*, *ORB*, *AKAZE*, *BRIEF*, and *BRISK*. Depending on the image conditions, each of these techniques has specific advantages and disadvantages. However, they all perform accurate matching of features across multiple images. In addition, we utilize a compositing technique to create seamless, high-quality panoramic images. This technique combines different frequency bands of the image to produce a smooth transition between overlapping regions. We evaluate the performance of the different feature descriptors based on the mean squared error (*MSE*), peak signal-to-noise ratio (*PSNR*) and processing time.

II. Image Stitching Techniques

The field of image stitching involves combining multiple images into a single, high-resolution image. This can be achieved through various techniques, including *feature-based* and *direct* image stitching. *Feature-based* image stitching involves identifying and matching key features in multiple images, while *direct* image stitching involves directly aligning and blending the images. Both approaches have their own advantages and challenges, and the choice of technique often depends on the specific application and requirements. In this chapter, we will explore the key concepts and techniques involved in *feature-based* and *direct* image stitching.

A. Feature-Based Image Stitching

Feature-based image stitching is a popular technique for combining two or more overlapping images into a single panoramic image. The basic idea is to identify corresponding features in each image and then use these features to align the images and blend them together. The feature detection and extraction algorithm is used to find the keypoints in the images. A popular algorithm used for this purpose is the Scale-Invariant Feature Transform (*SIFT*) algorithm. *SIFT* identifies keypoints in an image by analyzing the local gradient of the image and generating a set of scale-invariant descriptors that describe each keypoint. The keypoints are then matched across different images using feature matching techniques such as the *FLANN* matcher. [4] Once the keypoints are matched, a transformation matrix is calculated to align the images. This transformation matrix is calculated using the *RANSAC* algorithm, which identifies the best set of matches that can be used to estimate the

transformation. The transformation matrix is then used to warp the images so that they are aligned. Finally, the aligned images are blended together to create a seamless panoramic image. This can be done using a variety of blending techniques such as linear blending, multi-band blending, or feather blending.

B. Direct Image Stitching

Direct image stitching is an alternative approach to feature-based image stitching. In this technique, the images are directly aligned and blended together without the need for feature detection and matching. The basic idea behind direct image stitching is to align the images using a homography matrix. A homography matrix is a 3x3 transformation matrix that describes the relationship between two images. Once the homography matrix is calculated, it can be used to warp one image so that it aligns with the other image. To calculate the homography matrix, direct image stitching algorithms use a variety of techniques such as correlation matching, phase correlation, or optical flow. These techniques analyze the image intensities and calculate the transformation matrix that aligns the images. Once the images are aligned using the homography matrix, they can be blended together using the same techniques as feature-based image stitching. This creates a seamless panoramic image that combines the information from both images.

III. Image Stitching using Feature-Based Approach

The *feature-based* approach to image stitching involves identifying and matching key features in multiple images and then combining them into a single, seamless image. The process involves several key steps, in particular images acquisition, feature detection and description, image matching, and blending and composition (see Fig. 1). [8] In this chapter, we will explore each of these steps in detail, and discuss some of the key techniques and algorithms involved in the feature-based approach to image stitching.

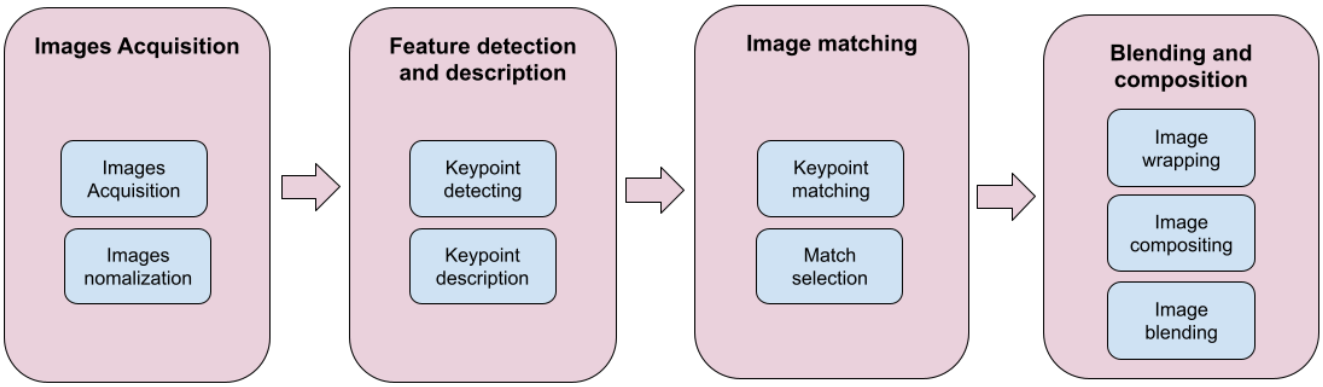

Fig. 1: The general steps of image stitching algorithms

A. Image Acquisition

In this section, we discuss the process of acquiring the images that will be stitched together. We obtain the images from different sources such as online datasets and our own custom images captured using a smartphone camera. Once the images are obtained, they are loaded into the program using the *OpenCV* library. These images may have different color spaces and intensity ranges, which can cause problems during the stitching process. Therefore, we normalize the intensity ranges of the images by scaling the pixel values to a range of 0-255 using the maximum pixel value of each image. We also convert the images to grayscale since most feature detection and extraction algorithms work with grayscale images. This is done using the *OpenCV* library's *cvtColor* function. Overall, the goal of the image acquisition and pre-processing step is to ensure that the images have consistent color and intensity characteristics, which makes it easier to detect and match features between the images.

B. Feature Detection and Description

1) Harris Corner: The *Harris corner* detector algorithm is a popular method used to detect and extract corners in digital images. It works by analyzing local changes in intensity in small image regions and identifying points where there are large variations in gradient. The corner detection process involves computing a Harris response matrix for each pixel in the image. This matrix contains information about the local gradient changes and is used to identify corner points. To identify corner points, a threshold value is applied to the Harris response matrix, and points with a response value greater than the threshold

are considered as corners. The threshold value is specified by the user and can be adjusted to increase or decrease the number of detected corners. To further improve the accuracy of the corner detection, a dilation step is applied to the Harris response matrix. This step helps to merge nearby corner points and eliminate noise. Finally, the corner points are converted into key points using the *cv2.KeyPoint* function, which takes the x and y coordinates of the corner point as input. These key points are stored in a list and returned by the function (see Fig. 5).

2) SIFT: Scale-invariant feature transform (*SIFT*) detects the interest points by blurring the images by convolution with different Gaussian kernels. From subtracting these blurring Gaussian kernels, the Difference of Gaussian kernel (*DOGs*) are generated. Based on *DOGs*, the interest points are found by an extreme kernel. Secondly, after collecting all the potential interest points, *SIFT* computes the gradient direction of each pixel of the area around the interest points, then builds a histogram of the gradient directions. The histogram is the description of interest points and is used later for matching with the interest point of other images. Only the direction of the gradient is counted, the magnitude of the gradient can be easily affected by the light condition or the quality of the camera (see Fig. 6).

3) ORB: In our report, we also utilized the Oriented *FAST* and Rotated *BRIEF* (*ORB*) feature descriptor in the image matching step of our feature-based image stitching algorithm (see Fig. 7). *ORB* is a feature descriptor that combines the strengths of two popular descriptors, Features from Accelerated Segment Test (*FAST*) and Binary Robust Independent Elementary Features (*BRIEF*). The *FAST* detector is used to detect keypoints in an image, while *BRIEF* is used to describe the local features around these keypoints. *ORB* is efficient and robust, making it a suitable choice for large-scale image stitching applications. It uses binary features to perform matching, which is faster and more efficient than traditional feature descriptors like *SIFT* and *SURF* that use floating-point features. ORB also has the advantage of being rotation-invariant, meaning it can detect and describe features regardless of their orientation in the image. This makes it more robust to changes in viewpoint and lighting conditions. [1]

4) KAZE: One of the feature descriptors that we implement is the Accelerated Kernelized Exhaustive Search (*KAZE*) descriptor, which is known for its robustness and efficiency (see Fig. 10 and Fig. 11). *KAZE* is a suitable alternative to traditional feature descriptors and can handle large variations in illumination and scale. It operates by computing a scale-space representation of the input images, followed by detecting and describing the key points using a nonlinear diffusion process. Our evaluation of different feature descriptors shows that *KAZE* performs well in terms of accuracy and processing time, highlighting its potential as a suitable choice for feature-based image stitching applications. [11]

5) FREAK: Fast Retina Keypoint (*FREAK*) is a binary feature descriptor that aims to provide a fast and efficient solution for image feature detection and matching. It operates by convolving a set of filters with the input image to create a multi-scale representation of retina-like features. These features are then transformed into a binary form, allowing for fast computation of feature matching across multiple images. Our evaluation of different feature descriptors reveals that FREAK exhibits high performance in terms of processing time and accuracy.

6) BRISK: Binary Robust Invariant Scalable Keypoints (*BRISK*) is a binary descriptor that is designed to be both fast and robust to image transformations, such as rotation and scaling. We evaluate the performance of the *BRISK* feature descriptor in the image matching step of our feature-based image stitching algorithm. It is based on the detection of scale-space extrema and can detect key points in highly textured or low-textured regions of an image. [13]

7) BRIEF: Binary Robust Independent Elementary Features (*BRIEF*) is a popular binary feature descriptor that has been designed to be both efficient and robust. In *BRIEF*, pairs of pixels are randomly sampled from the image, and a binary code is computed based on their intensity comparison. This binary code is then used to describe the features of the image. Compared to other feature descriptors, *BRIEF* is computationally efficient and is well suited for real-time applications. However, *BRIEF* does have some limitations in terms of its robustness to noise and illumination changes, which can lead to a decrease in performance. To address these limitations, *BRIEF* is often combined with other techniques such as scale-invariant feature transform (*SIFT*) or oriented *FAST* and rotated *BRIEF* (*ORB*) to enhance its performance in feature-based image stitching applications. Despite its limitations, *BRIEF* has been widely used in various computer vision applications, including object recognition, image classification, and feature-based image stitching. [7]

8) AKAZE: Accelerated-KAZE (*AKAZE*) is a robust feature descriptor that is designed to be both scale and rotation-invariant. It uses a nonlinear scale space that adapts to the local structure of the image, making it suitable for applications where the image scale varies significantly. *AKAZE* also includes a novel method for assigning orientations to feature points, which helps improve the accuracy of the feature-matching process.

C. Image Matching

1) Keypoint matching: We utilize two kinds of matcher, Brute-Force matcher (*BF* matcher) and Fast Library for Approximate Nearest Neighbors matcher (*FLANN* matcher). The *BF* matcher is simple. It evaluates each distance measurement between the keypoint descriptions of two images and identifies the optimal matches. However, *BF* matcher requires more time and computing resources. The *FLANN* matcher, on the other hand, is a quick and effective algorithm for approximate nearest neighbor search. It applies a hierarchical tree structure to efficiently search a high-dimensional space for the nearest neighbors.

In this issue, we use the *FLANN* matcher for *SIFT* and *KANE*. The *BF* matcher is applied for the rest.

2) Matching selection: There are two types of matching selection used in this process: ”original” and *KNN* matching. In the ”original” matching mode, key points are matched based on the Euclidean distance between their descriptors. In the *KNN* matching mode, the K-Nearest Neighbors algorithm is used to find the best matching pairs of descriptors, and the Lowe's ratio test is applied to filter out unreliable matches. Finally, the matched key points are returned in the form of arrays containing the coordinates of corresponding points.

D. Blending and Composition

After obtaining the homography matrix for the input images, the next step is to stitch them together and preserve the format of the original images. *Feature-based* image stitching requires blending to make seamless and lifelike panoramic photos. The first step is wrapping one of two images with a transformation to the frame of the other. However, the transformation has a problem, which is it can expand or shrink the images. Therefore, to avoid the original images getting cropped, there is a need for another transformation to expand the padding of the whole image. Then, we can composite the wrapped images into the second image to have the panorama. The connection parts are blended in together, because each picture can be taken under different illumination.

IV. Result and Analysis

Our results of applying the different methods are reported in the Table I. The results are evaluated in terms of two metrics, mean squared error (*MSE*) and peak signal-to-noise ratio (*PSNR*). *BRIEF* has the highest *MSE* value of 93.65, indicating that it performs better in terms of minimizing the difference between the original data and the processed data. However, it has a relatively low *PSNR* value of 9.98, suggesting that the processed data may not be as faithful to the original data as some of the other methods. Looking at the speed, the fastest performer is *ORB* and *BRIEF*, whereas *KAZE* is the slowest being nearly four times slower than the fastest. The choice of method should take into account the specific application and the trade-offs between speed, accuracy, and quality. The setting up for experiments in this table is based on the first two images in *”xue-mountain”* folder (see Fig. 10). The performance values are an average of 50 runs on a system with the CPU Ryzen 5900X using Python 3.11 on Windows 11.

TABLE I: Measurement of image quality and performance resulting from feature-based image methods. The results are based on the first two images of the dataset *xue-mountain*.

Method	MSE	PSNR	Speed	
			in seconds	in %
SIFT	91.26	10.94	0.44	57.55
ORB	92.25	10.09	0.25	100
KAZE	91.13	10.16	0.94	26.6
FREAK	92.58	9.81	0.27	92.59
BRISK	91.74	10.14	0.41	60.98
BRIEF	93.65	9.98	0.25	100
AKAZE	92.85	10.01	0.40	62.5

We apply the feature-based technique for various panorama images, and different steps of our implementation are shown in the below figures.

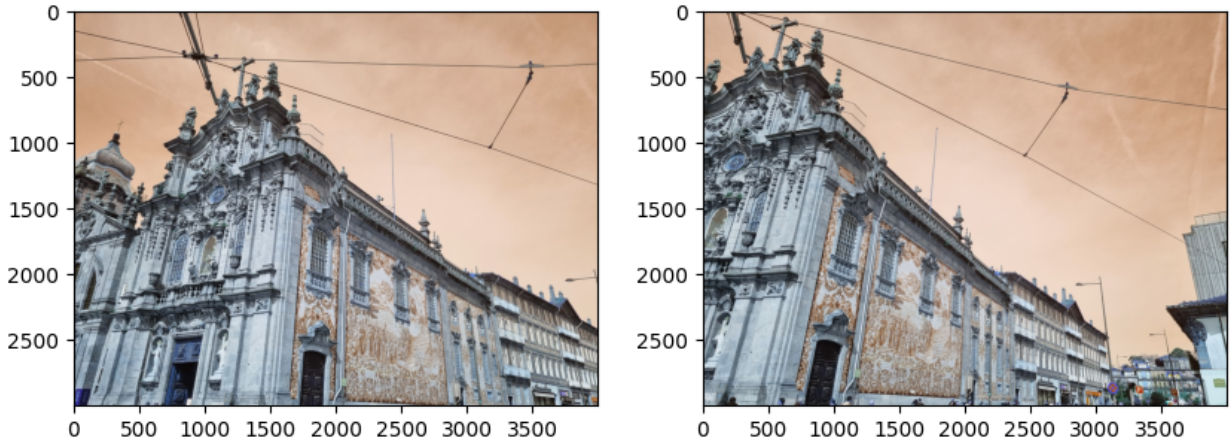

Fig. 2: *Original images*

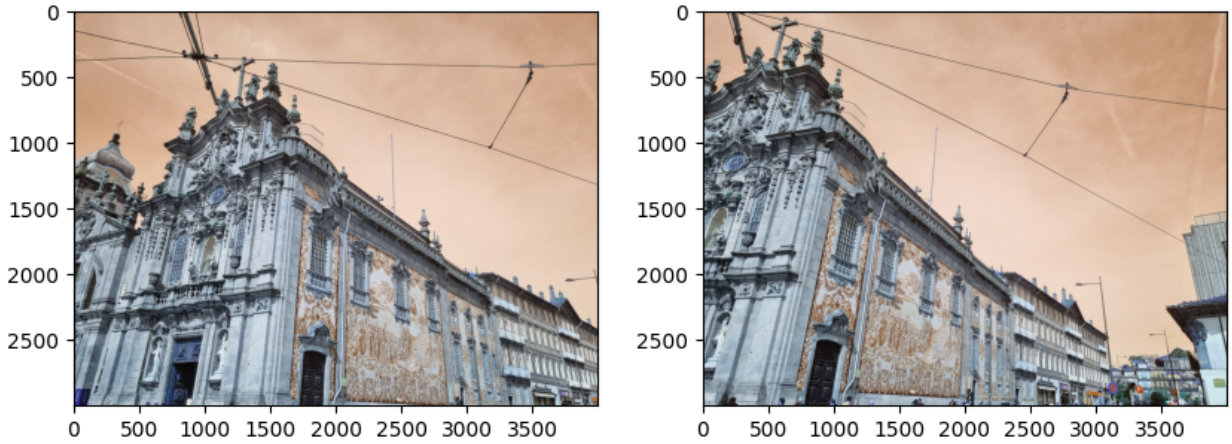

Fig. 3: *Nomalized images*

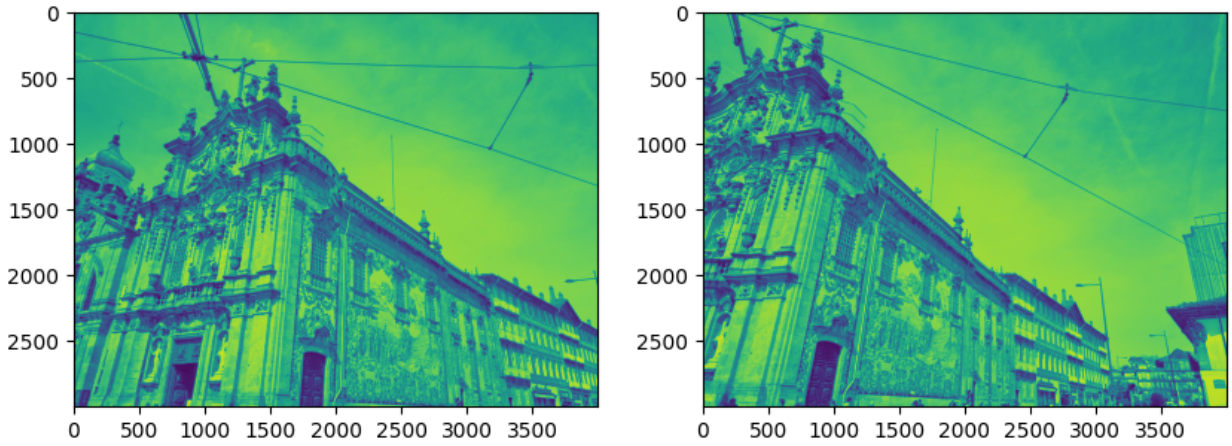

Fig. 4: *Gray images*

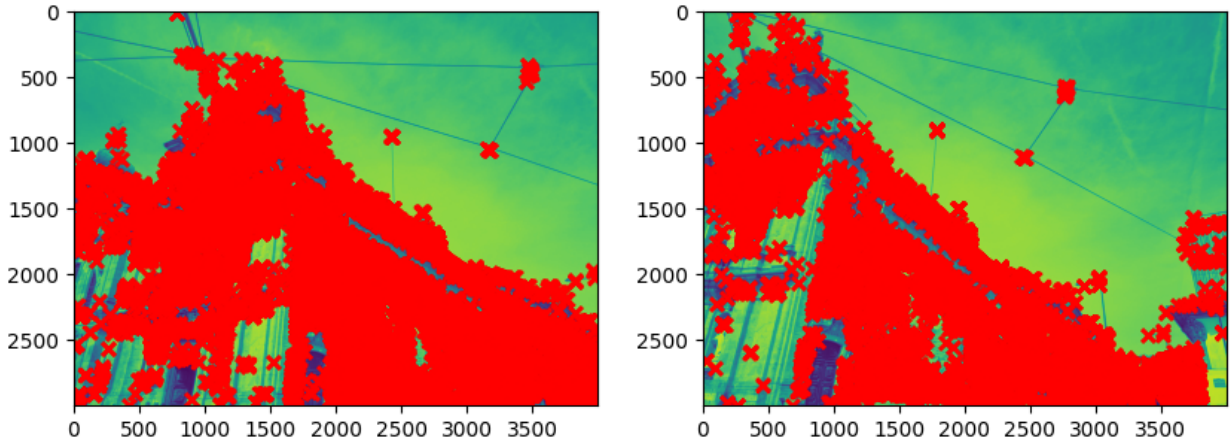

Fig. 5: *Keypoints detected by **Harris corner** dectector*

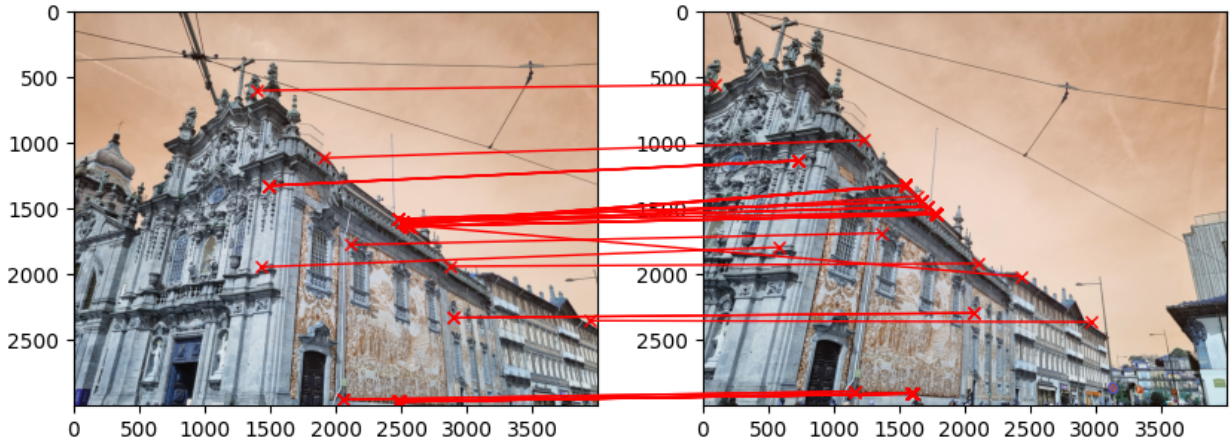

Fig. 6: *Keypoints matching by **SIFT** descriptor and **FLANN** matcher*

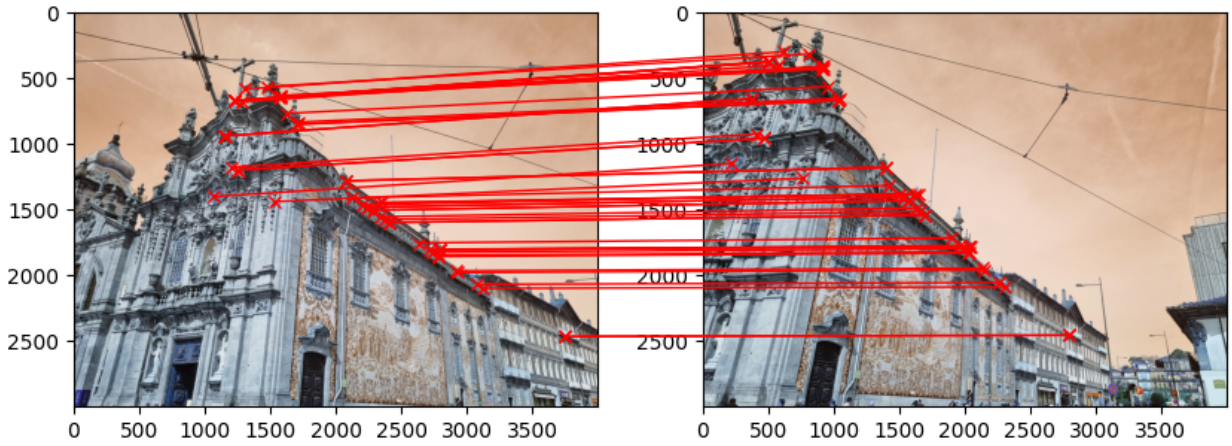

Fig. 7: *Keypoints matching by **ORB** descriptor and **BF** matcher*

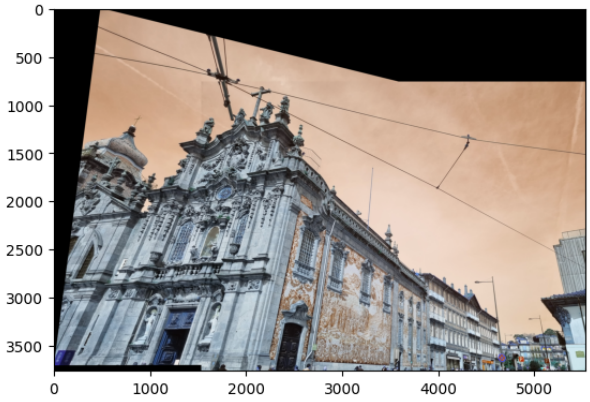

Fig. 8: *Panorama of two images with **SIFT** and **Harris corner***

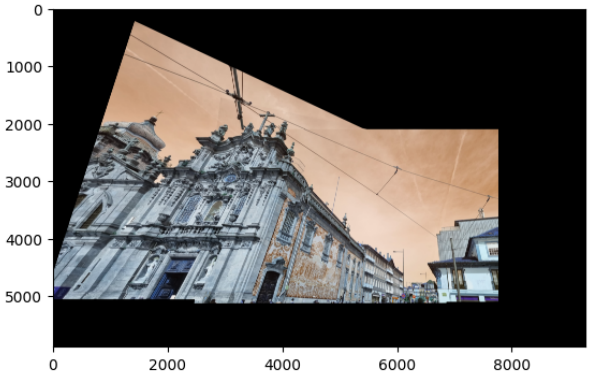

Fig. 9: *Panorama of three images with **SIFT** and **Harris corner***

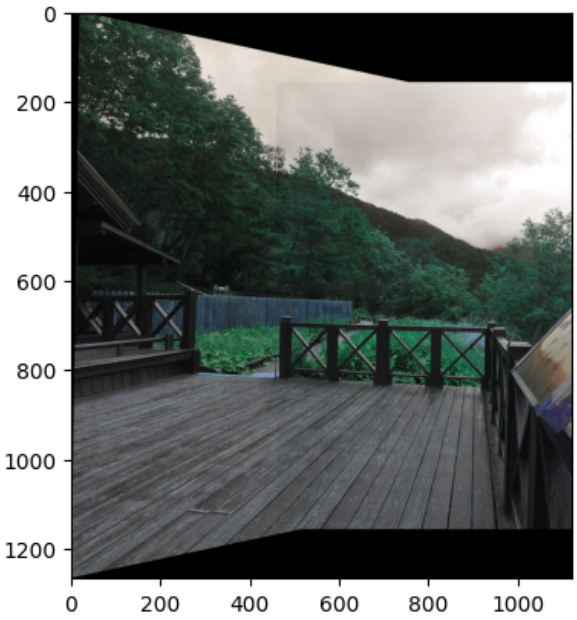

Fig. 10: *Panorama of two images with **KAZE***

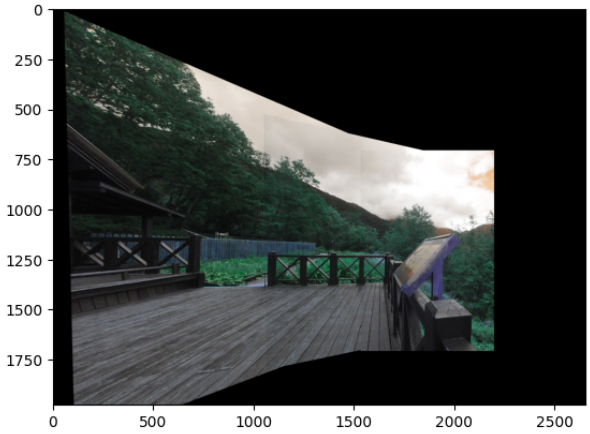

Fig. 11: *Panorama of three images with **KAZE***

V. Challenges of Feature-based Image Stitching

Feature-based image stitching comes with its own set of challenges. One of the primary challenges with *feature-based* image stitching is the accuracy and robustness of feature detection and matching algorithms. These algorithms need to be able to identify and match features across images that may have significant differences in lighting, color, and texture. Another challenge is handling non-linear perspective distortions, such as those caused by wide-angle lenses or complex scenes. In addition, *feature-based* image stitching requires significant computational resources, making it computationally expensive and time-consuming. Furthermore, when dealing with repetitive landscape images such as those captured at sea (see Fig. 12), where the images contain similar textures and structures. In such cases, feature detection and matching algorithms may struggle to identify and match the correct features across the images, resulting in errors in the final stitched panorama [5].

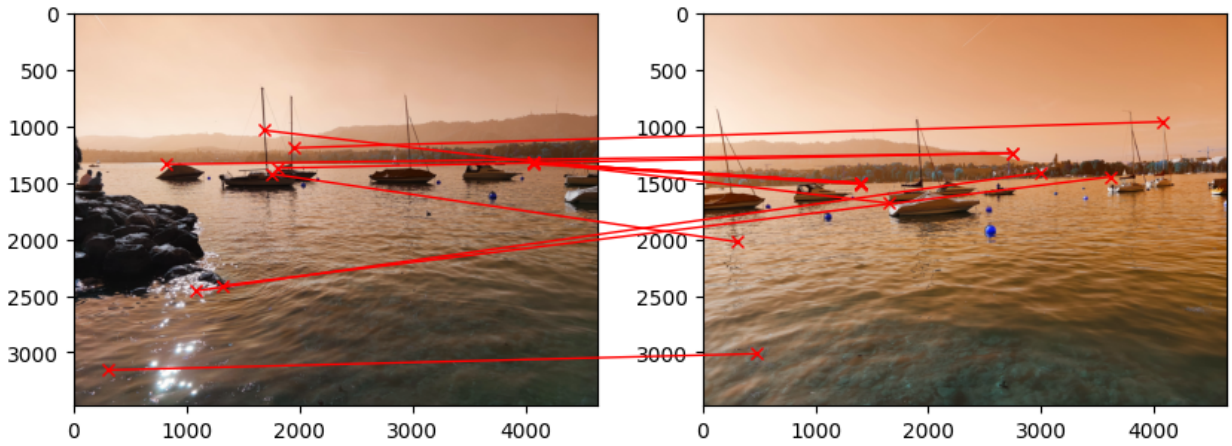

Fig. 12: ***SIFT*** *detector wrongly describes the differences when there are a lot of key points which have almost the same gradient direction*

VI. Conclusion and future work

In conclusion, *feature-based* image stitching is a powerful technique for creating panoramic images and high-resolution images by combining multiple images with overlapping fields of view. This paper has explored different feature-based image stitching descriptors like *Harris*, *SIFT*, *KAZE*, *ORB*, *AKAZE*, *BRIEF*, and *BRISK*. They are evaluated on their performance using various metrics such as processing time, *MSE*, and *PSNR*. Although there are still some inaccurate results in some cases, our evaluation of various feature descriptors has demonstrated that it is capable of stitching images in the majority of instances in terms of both accuracy and processing time, making it a promising option for large-scale image stitching applications. We found that *BRIEF* performed the best overall in terms of both accuracy and processing time, making it a promising choice for large-scale image stitching applications. In the future, the use of deep learning techniques for feature extraction and matching should be explored to further enhance the performance of feature-based image stitching algorithms.

REFERENCES

[1] Kurt Konolige Ethan Rublee, Vincent Rabaud and Gary Bradski. Orb: an efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, 2011.

[2] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Association for Computing Machinery*, 24(6), 1981.

[3] Mingxiao Huo, Zhihao Zhang, and Xianqiang Yang. Deep image stitching with pixel similarity correlation. In *2022 China Automation Congress (CAC)*, pages 4316–4321, 2022.

[4] David G. Lowe Marius Muja. Fast approximate nearest neighbors with automatic algorithm configuration. *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009.

[5] David G. Lowe Matthew Brown. Recognising panoramas. *Proceedings Ninth IEEE International Conference on Computer Vision*, 10 2003.

[6] David G. Lowe Matthew Brown. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 12 2006.

[7] Christoph Strecha Pascal Fua Michael Calonder, Vincent Lepetit. Brief: Binary robust independent elementary features. 2010.

[8] Mohammad Shorif Uddin Moushumi Zaman Bonny. Feature-based image stitching algorithms. *2016 International Workshop on Computational Intelligence (IWCI)*, 2016.

[9] Lang Nie, Chunyu Lin, Kang Liao, Shuaicheng Liu, and Yao Zhao. Unsupervised deep image stitching: Reconstructing stitched features to images. *IEEE Transactions on Image Processing*, 30:6184–6197, 2021.

[10] Lang Nie, Chunyu Lin, Kang Liao, Shuaicheng Liu, and Yao Zhao. Deep rectangling for image stitching: A learning baseline. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5730–5738, 2022.

[11] Andrew J. Davison Pablo Fernández Alcantarilla, Adrien Bartoli. Kaze features. 2013.

[12] Mikail Deniz Cayoglu Sima Esmaeili, Ha Van Dang. Github repository github.com/Raichuu41/3d-cv-image-stitching. 2023.

[13] Roland Y. Siegwart Stefan Leutenegger, Margarita Chli. Brisk: Binary robust invariant scalable keypoints. *2011 International Conference on Computer Vision*, 2011.

[14] Richard Szeliski. Image alignment and stitching: A tutorial. 12 2006.

ACKNOWLEDGMENT

The authors would like to thank Prof. Rother for the engaging lecture.

SOURCE CODE

The code can be found in the GitHub repository. [12]

VII. CONTRIBUTION

We worked equally as much on each topic. We used a lot of peer-programming and nightly meetings to create, discuss, fine-tune and review each other's work input.