VR Assignment I
Report

Submitted By : Mohammed Danish Rabbani [MS2024506]

Repository Link

1 Introduction

This report presents the implementation details of two VR assignment tasks: Coin Detection and Panorama Stitching.

- Coin Detection and Segmentation: Detect, segment, and count Indian coins from an image using image processing techniques.
- Panorama Stitching: Create a stitched panorama from multiple overlapping images using keypoint detection and image alignment.

2 Coin Detection and Segmentation

2.1 Objective

- **Process Edges** - To explore various techniques to identify and extract the edges in the images of coins
- **Segmentation and coin count** - To explore various techniques to segment out each coin from the image and count the number of coins

2.2 Implementation

2.2.1 Edge Detection

Multiple images have been use to showcase how edges can be detected using the following techniques -

1. Sobel Operator
2. Prewitt Operator
3. Canny Edge Detection

In order to improve the edge detection , gaussian blurring was used. Since the amount was blurring needed (kernel size) differs from image to image , an adaptive gaussian blurring technique was implemented where the kernel size is selected based on laplacian variance.

2.2.2 Edge Based Segmentation

The Canny edge detection method is employed to extract prominent edges, which are then dilated using morphological operations to strengthen the contours. The **cv2.findContours** function retrieves the external contours, filtering out small areas to eliminate noise. Finally, valid contours are drawn on the original image, and individual segmented coin regions are extracted using a mask. The method effectively isolates coin objects based on their edges and ensures precise segmentation by leveraging both edge detection and contour-based filtering. The number of segmented coins is determined by counting the extracted contours.

2.2.3 Region Based Segmentation

The process begins with converting the input image to grayscale, followed by adaptive Gaussian blurring to reduce noise while preserving structural details. The Otsu's thresholding method is then applied to automatically determine an optimal threshold value for binarization. Depending on the mean intensity of the grayscale image, either a binary or inverted binary threshold is selected to ensure effective segmentation.

To refine the segmentation, morphological opening is performed using a small kernel to remove noise and smoothen the binary mask. The external contours of the thresholded image are then extracted using cv2.findContours, filtering out small regions based on area constraints to eliminate noise and irrelevant objects. A binary mask is created for each valid contour, allowing for precise extraction of individual coins. Finally, the contours are drawn on the original image for visualization.

2.2.4 Count coins

Number of contours with an area about a given threshold are considered as coins.

2.3 Observation

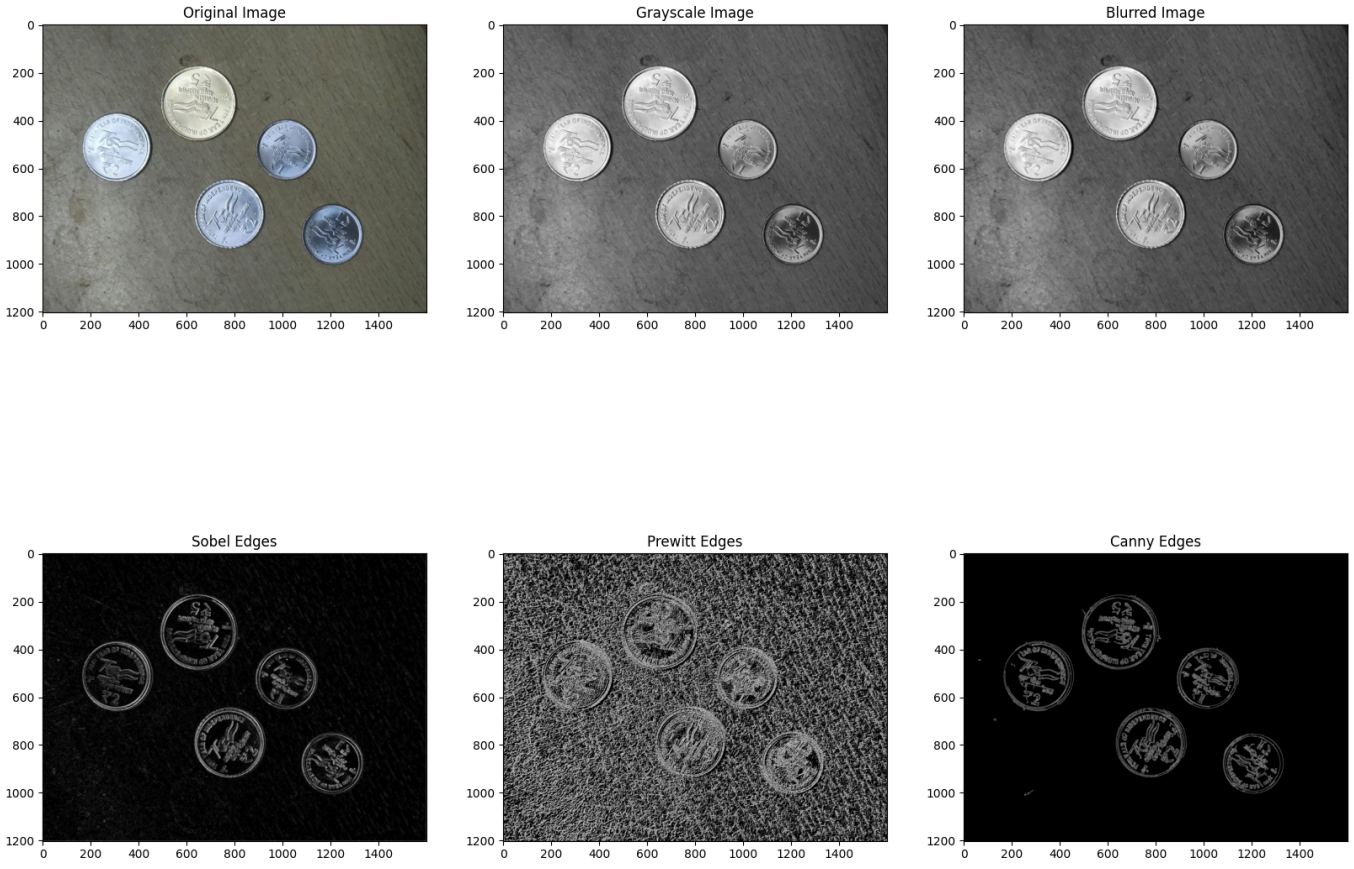
A total of 13 images had been captured and used in the assignment. Here are some observations.

- Each images requires different preprocessing depending on various parameters like brightness, contrast etc
- For some images the Edge based segmentation technique works better than Region based segmentation - Mainly images with varying brightness
- For some images the Region based segmentation technique works better than edge based segmentation - Mainly images with not varying brightness

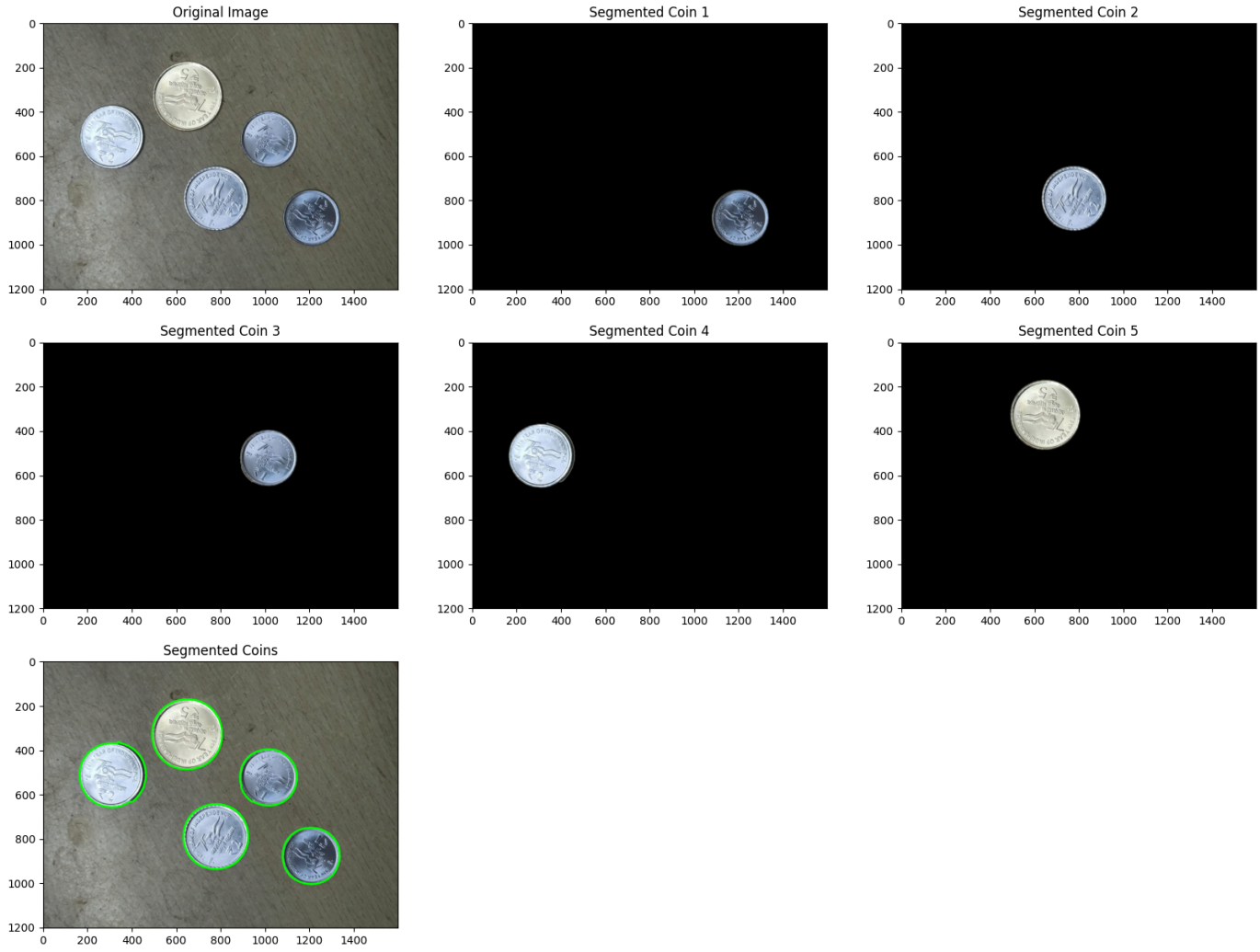
- Canny Edge detection performs poorly on images with texture backgrounds.

2.4 Results

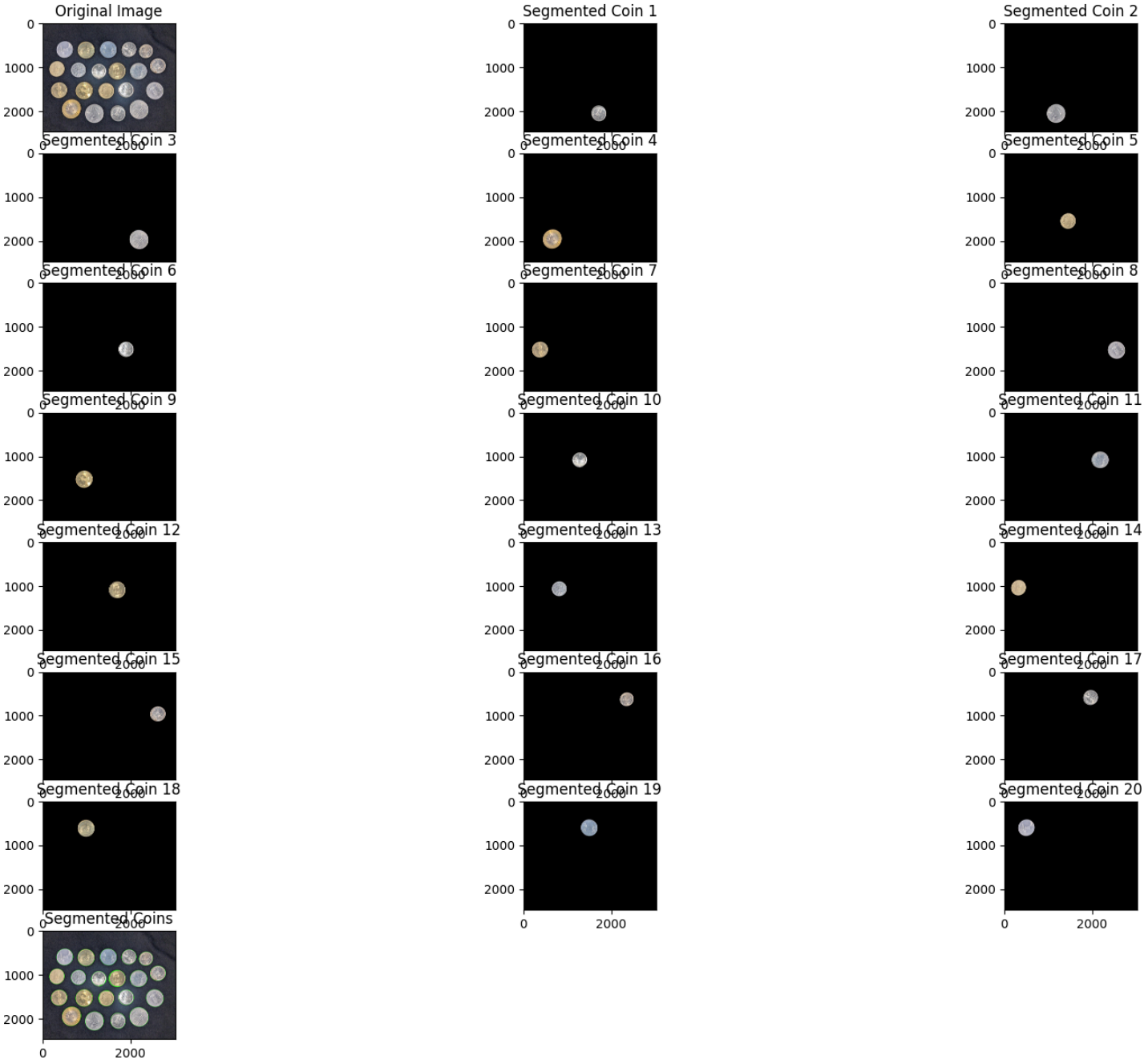
2.4.1 Edge Detection



2.4.2 Edge Based Segmentation



2.4.3 Region based Segmentation



3 Panorama Stitching

3.1 Objective

The objective of the panorama stitching task is to combine multiple images into a seamless panoramic image. The process involves keypoint extraction, feature matching, homography estimation, and image warping

3.2 Implementation

3.2.1 Image Preprocessing

- The input images are converted to grayscale using OpenCV's cv2.cvtColor() function. Grayscale conversion simplifies feature detection by reducing computational complexity.

3.2.2 Feature Detection using SIFT

- The SIFT algorithm is applied to detect keypoints and compute feature descriptors in each image.
- The function sift.detectAndCompute() extracts distinctive features that are robust to scale, rotation, and illumination changes.

3.2.3 Feature Matching using FLANN

- FLANN-based matching is used to find correspondences between feature descriptors of the two images.
- The K-nearest neighbor (KNN) algorithm is applied to find the top two matches for each feature.
- The Lowe's ratio test is performed to filter good matches, ensuring only the most relevant feature correspondences are retained.

3.2.4 Homography Transformation

- If a sufficient number of feature matches are found, the homography matrix is estimated using cv2.findHomography().
- The RANSAC (Random Sample Consensus) algorithm is applied to handle outliers and improve accuracy.
- The first image is warped onto the second image using cv2.warpPerspective() based on the computed homography matrix.

3.2.5 Image Blending and Stitching

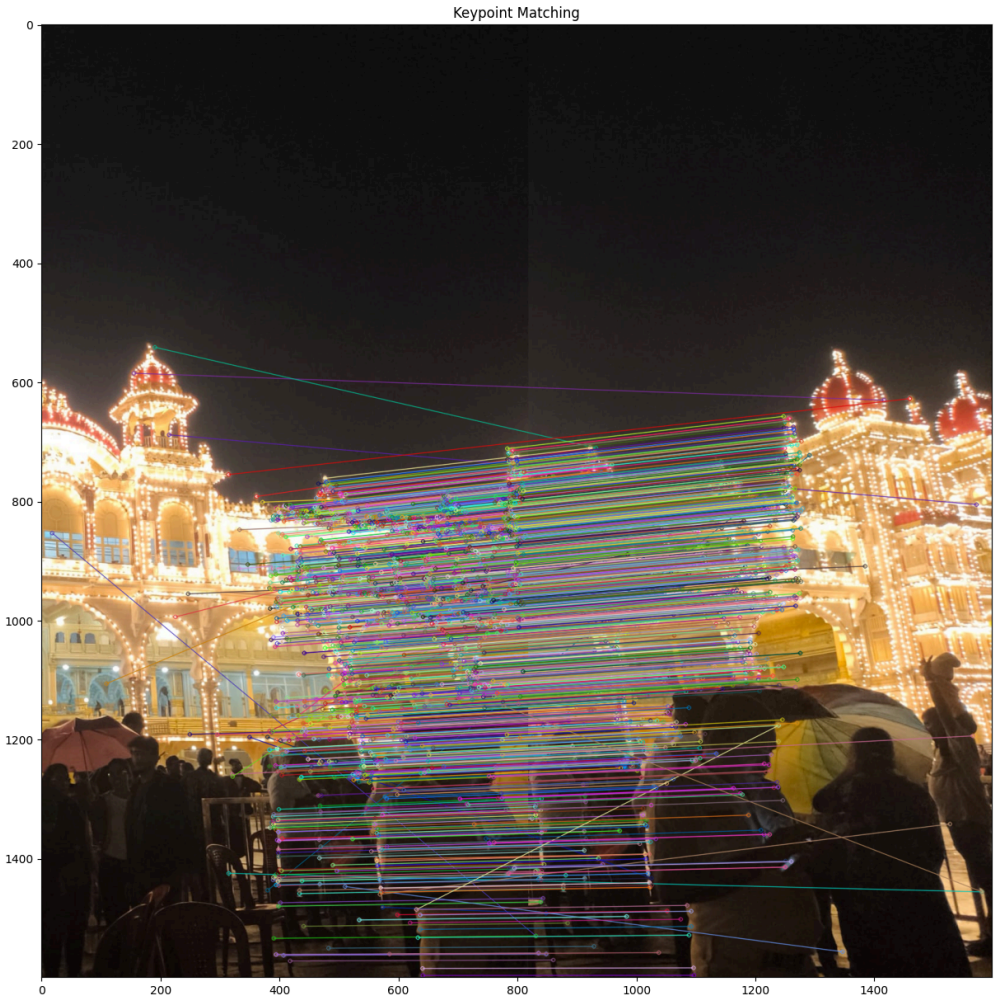
- The warped image is merged with the reference image.

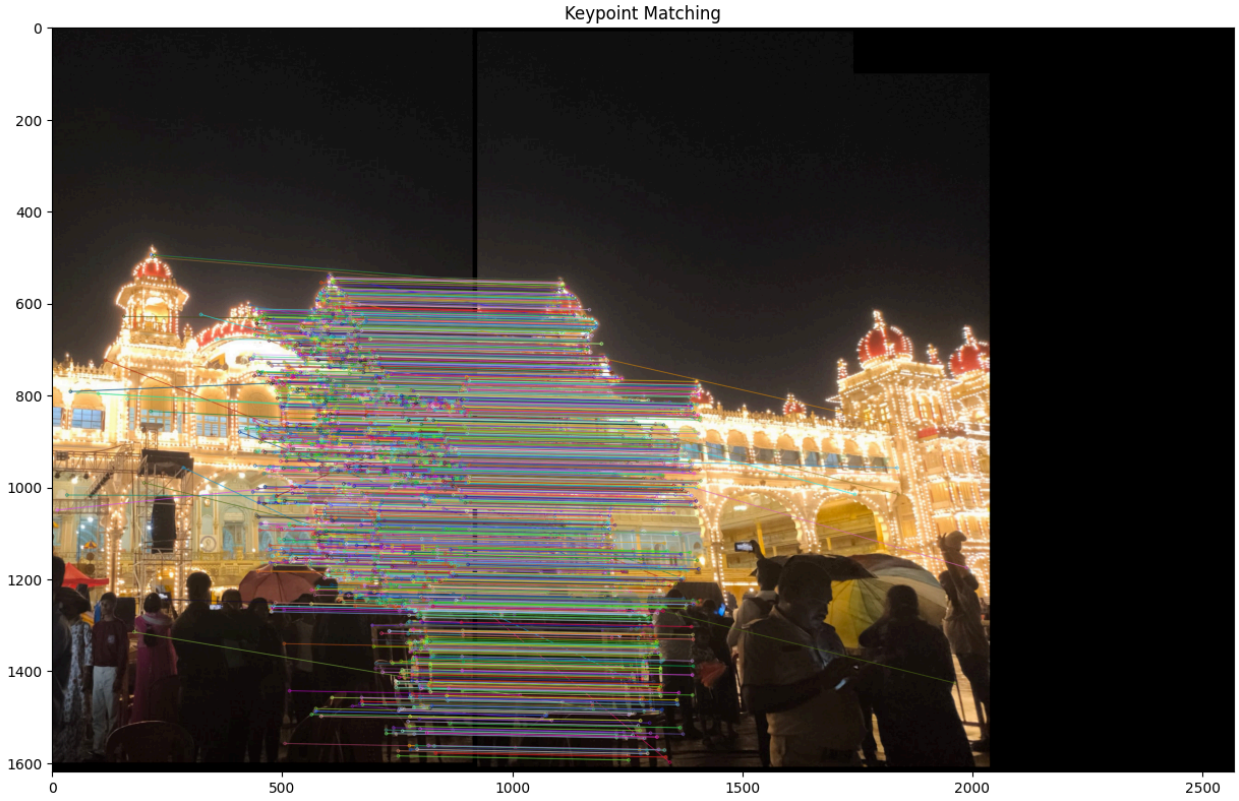
- Border extension is applied to avoid artifacts using cv2.copyMakeBorder().

3.3 Observations

- The implementation successfully aligns and stitches images to create a panoramic view.
- The SIFT feature detector ensures reliable feature extraction, while FLANN-based matching accelerates the process.
- The use of homography transformation with RANSAC helps in achieving accurate alignment even in the presence of outliers.
- The output panorama is visually coherent, with minimal distortions in well-aligned cases.

3.4 Results




Keypoint Matching
0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000
2500


Keypoint Matching
0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000
2500

3.4.1 Final Image