VR Assignment 1

Vedant Mangrulkar

February 23, 2025

1 Coin Detection

1.1 Image Preprocessing

The input images are loaded and resized for consistency. To enhance feature detection, grayscale conversion and Gaussian blurring are applied. The blurring reduces noise and improves object differentiation. Adaptive thresholding is then used to convert the image into a binary format for further processing.

1.2 Circular Object Detection

Contours are extracted from the binary image, and each detected shape's area and perimeter are analyzed. The circularity measure, defined as:

$$C = \frac{4\pi A}{P^2} \tag{1}$$

is used to classify objects as circular. Objects with a circularity close to 1 and an area above a certain threshold are identified as coins.

1.3 Contour and Segmentation Visualization

Detected coins are visualized using two techniques:

- **Contour detection:** Outlines of detected objects are drawn on the original image.
- **Segmentation:** A mask is created, allowing only detected coins to be visible while removing the background.

2 Results

The results of the coin detection algorithm are displayed below. The first image shows the detected contours, the second image shows the individual extracted coin and the third image shows the segmented coins.

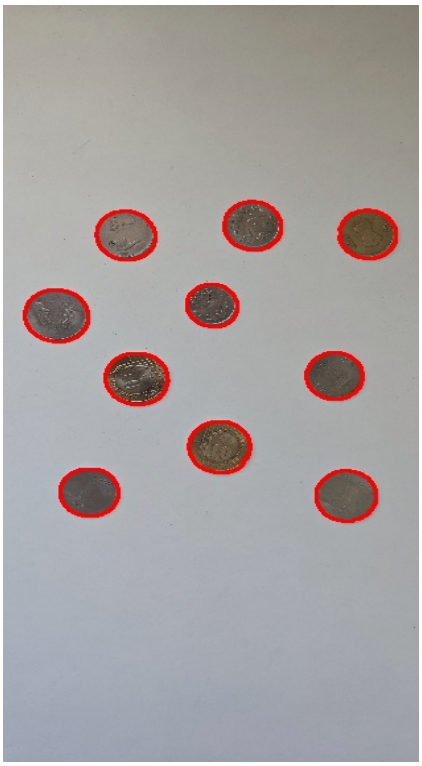

Figure 1: Detected contours of coins.

Figure 2: Individual extracted coin.

Figure 3: Segmented coins using mask processing.

3 Image Stitching

3.1 Image Rescaling

Each input image is resized to a standard width while preserving its aspect ratio to ensure consistency in feature detection.

3.2 Feature Extraction

Key points and descriptors are extracted using the **Scale-Invariant Feature Transform (SIFT)**. These key points serve as unique features that can be matched across overlapping images.

$$\text{Keypoints, Descriptors} = \text{SIFT.detectAndCompute(image)} \tag{2}$$

3.3 Feature Matching

A **Brute-Force Matcher (BFMatcher)** is used to find the best correspondences between key points in overlapping images. The Lowe's ratio test is applied to filter out weak matches.

$$\text{Good Matches} = \left\{ m_0 \mid \frac{m_0}{m_1} < \text{Threshold} \right\} \tag{3}$$

3.4 Homography Estimation

If enough good matches are found, a homography matrix H is computed using the **RANSAC (Random Sample Consensus)** algorithm. This transformation matrix allows one image to be warped onto another.

$$H = \text{findHomography}(Points_1, Points_2, \text{RANSAC}) \tag{4}$$

3.5 Image Warping and Merging

The left image is warped using the homography matrix, and the right image is placed on top to create a stitched panorama.

3.6 Cropping the Final Output

After stitching, the black borders introduced due to warping are removed by detecting the bounding region of the non-black pixels.

3.7 Implementation

The image stitching process is implemented using **OpenCV and NumPy**. The following steps summarize the key components:

- **Extract keypoints** using the SIFT algorithm.
- **Match keypoints** between adjacent images.
- **Compute the homography matrix** to align images.
- **Warp images** and blend them together.
- **Refine and crop the final stitched image** to remove black borders.

3.8 Results and Observations

The final output is a seamlessly merged image of the input images. The matching process ensures that overlapping regions align correctly, preserving structural integrity and visual quality.

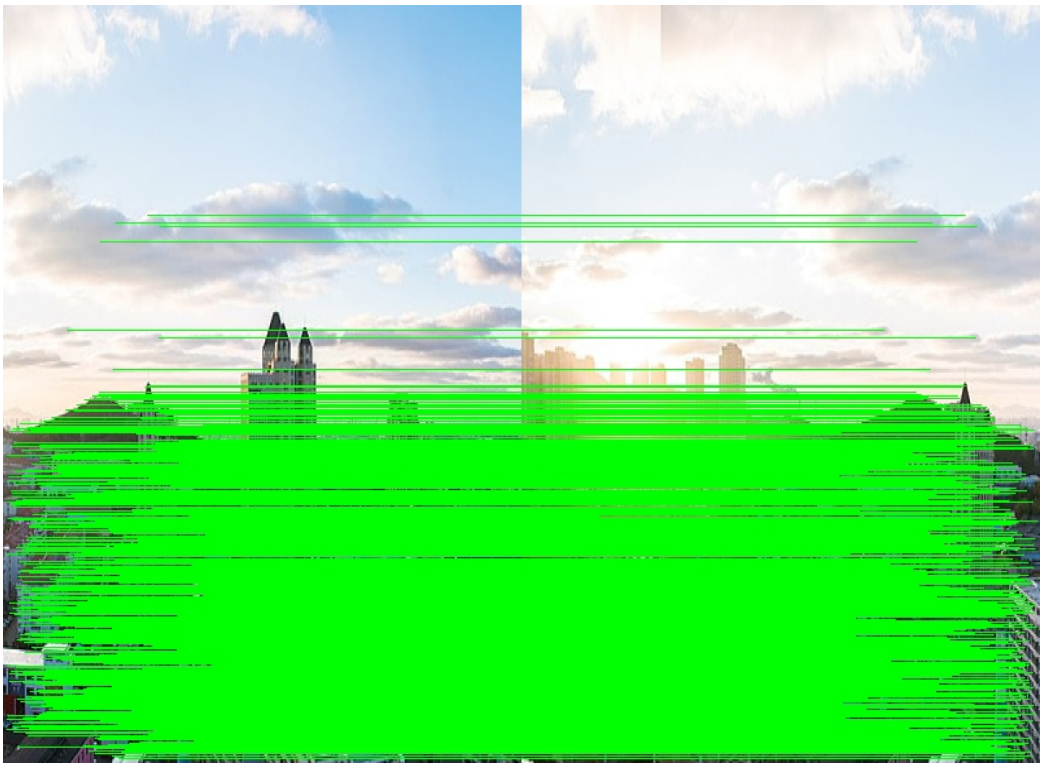

Figure 4: Matched descriptors of 2 of the used images.

Figure 5: Final stitched panorama image.

Please click the below github link for different input and output responses for both the parts along with their respective codes: **Github link**