VR Assignment - 1

Bhavya Kapadia
IMT2022095

March 1, 2025

1 Part 1 - Detection, Segmentation And Coins Count

1.1 Data Preprocessing

The image undergoes several preprocessing steps to enhance its quality for analysis. Initially, it is converted to grayscale to simplify its representation and emphasize intensity variations. The image is then resized while preserving the aspect ratio to ensure uniformity across different inputs. To improve contrast, Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied, which enhances details in darker regions. A Gaussian blur is then used to reduce noise and smoothen the image. Even a Low Pass Filter was tried, but resulted in less efficiency than Gaussian Blur. Following this, Otsu's binarization automatically determines an optimal threshold, segmenting the foreground from the background. Even Binary Adaptive Thresholding was implemented but led to detection of inner curves of thew coins. Finally, morphological closing is applied using a kernel to fill small gaps and refine object boundaries, making the image more suitable for further processing.

1.2 Coin Detection

For coin detection, I initially implemented edge detectors like Marr-Hildreth and Canny Edge Detector. However, these methods also detected the inner details of the coin, leading to incorrect counting. Despite adjusting their parameters, the issue persisted as inner curves were still being detected. To address this, I increased the image's smoothness before reapplying the same algorithms, which significantly reduced inner detections. Eventually, I discovered the `findContours` function in OpenCV (`cv2`), which proved to be far more effective than both edge detection methods. Circularity analysis is used to identify potential coins by checking how closely a detected contour resembles a perfect circle. The circularity is computed using the formula:

$$\text{Circularity} = \frac{4\pi \times \text{Area}}{\text{Perimeter}^2}$$

Contours with a circularity value between 0.5 and 1.5 are considered coin-like, helping to filter out irregular shapes. Additionally, the code ensures accurate detection by evaluating

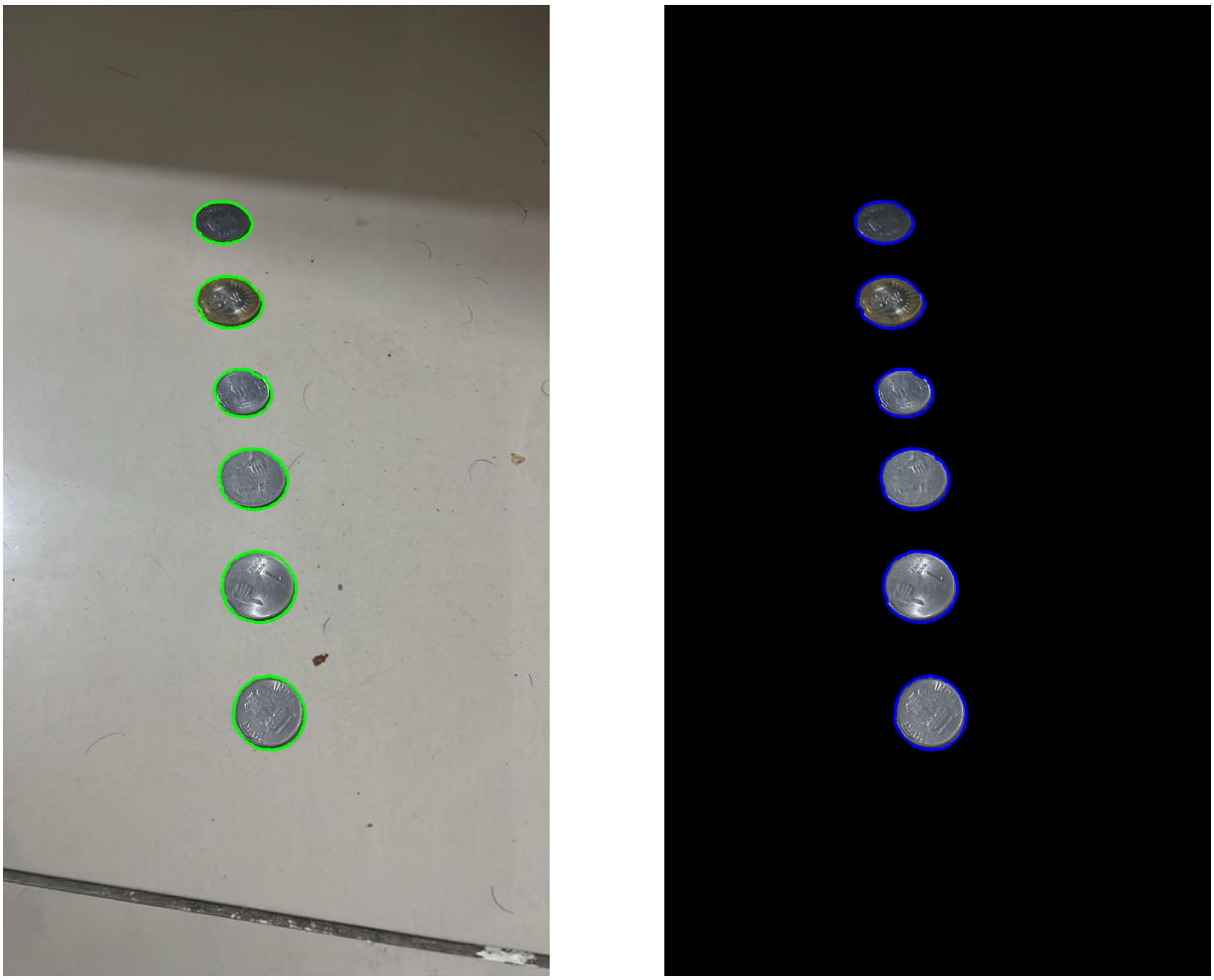
the axis ratio of the fitted ellipse. Only contours that meet all these shape constraints, including a perimeter threshold and area limits, are classified as coins.

Figure 1: Input Image

The detected coins for the input image above are shown in Figure 2:

1.3 Segmentation and Coin Counting

First, the binary mask is resized to match the original image dimensions. The contours are then scaled back using the aspect ratio to ensure accurate placement. For each detected coin, a separate mask is created, and cv2.drawContours is used to extract the coin region from the original image. The extracted coin is then cropped using its bounding box and saved as a separate image. Additionally, a combined mask is generated for visualization, highlighting detected coins by overlaying contours on a black background with blue outlines. This approach effectively isolates and saves each coin while providing a clear segmented



(a) Detected Coins

(b) Segmented Coins

Figure 2: Comparison of Detected and Segmented Coins

output for further analysis. The code written would return the number of coins as 6. The coins are counted as the size of the list wherein the objects were stored which passed the circularity check.

(a) Coin 1

(b) Coin 2

(c) Coin 3

(d) Coin 4

(e) Coin 5

(f) Coin 6

Figure 3: Segmented Coins

2 Part 2 - Stitched Panorama From Multiple Images

2.1 Data Preprocessing

The process begins with converting images to grayscale, reducing computational complexity while retaining structural details. To ensure consistent illumination and contrast, histogram equalization can be applied, enhancing the visibility of key features. Noise reduction techniques, such as Gaussian blurring, help suppress minor variations while preserving important edges.

2.2 Feature Extraction

SIFT (Scale - Invariant Feature Transform) is applied here to extract distinctive keypoints from an image by analyzing intensity variations, making it highly robust to changes in scale, rotation, and lighting. Each detected keypoint is assigned a descriptor that aids in identifying and matching similar points across overlapping images. By comparing these keypoints across multiple images, the algorithm ensures precise alignment of corresponding regions. This accurate feature matching facilitates seamless stitching, resulting in a well-blended and visually coherent panoramic image. The SIFT Detector detects and computes the key points and descriptors.

2.3 Detection and Visualization of Matches

The function helps match keypoints between two images and find the best way to align them. First, it uses Brute-Force Matching (BFMatcher) to compare keypoints from both images and find possible matches. It applies the K-Nearest Neighbors (KNN) algorithm, which finds the two closest matches for each keypoint. To remove weak matches, it uses Lowe's ratio test, keeping only the best matches where the first match is much better than the second one. If there are at least four good matches, the function extracts their positions and applies RANSAC (Random Sample Consensus) to calculate a homography matrix. This matrix helps determine how one image should be transformed to align with the other while ignoring incorrect matches. Finally, the function returns the filtered matches, the homography matrix, and a mask that highlights the best matches, making the image stitching process more accurate. The main issue I faced here was getting a proper drawing of the matches which were different because the different sizes of the images and less overlapped region. To tackle this, I resized the images at every instance and then the overlapped region was also increased and then image stitching was done more accurately.

2.4 Merging of the Images

The function combines two images using a homography matrix to align them properly. It first finds the sizes of both images and calculates how the first image's corners will be transformed. Based on this, it determines the new canvas size to fit both images without cutting anything off. Since some parts may shift out of view, it adjusts the position using a translation matrix to keep everything within the visible area. Then, the first image is warped to match the

perspective of the second image. Finally, the second image is placed on the canvas in the correct spot, creating a smoothly stitched final image.

(a) Input 1 (b) Input 2

Figure 4: Input Images 1 and 2

Subsequently two more inputs were provided and then two more image stitching was done before the final panorama is created.

Figure 5: Match Step 1

(a) Input 3

(b) Input 4

Figure 6: Input Images 3 and 4

Figure 7: Match Step 2

Figure 8: Match Step 3

Figure 9: Final Panorama