AIM 825 Visual Recognition: Assignment 1

Praveen Peter Jay
IMT2022064

March 1, 2025

1 Introduction

This report documents the implementation and results of two computer vision tasks: coin detection and image stitching. The goal was to apply image processing techniques to detect coins in images and stitch multiple images into a panorama.

2 Coin Detection and Counting

2.1 Approach

The coin detection algorithm processes images as follows:

1. Convert the input image to grayscale to simplify processing.
2. Apply Gaussian blur to reduce noise and smooth the image.
3. Use edge detection techniques to identify boundaries of coins.
4. Find and draw contours around detected objects.
5. Segment individual coins and save them as separate images.
6. Count the number of detected coins.
7. Verify that the detected objects are coins using shape analysis and size filtering.

2.2 Edge Detection Techniques

Edge detection is crucial in identifying coin boundaries. Three different methods were applied:

- **Sobel Operator:** Computes the gradient magnitude in the x and y directions, emphasizing regions with high-intensity changes.
- **Laplacian Operator:** A second-order derivative method that detects rapid intensity variations.
- **Canny Edge Detector:** A multi-stage algorithm involving Gaussian filtering, gradient intensity computation, non-maximum suppression, and hysteresis thresholding for optimal edge detection.

2.3 Object Identification: Distinguishing Coins

To differentiate coins from other objects, the algorithm applies:

- **Circular Hough Transform (CHT):** The Circular Hough Transform is used to detect circular objects within the image. This method works by transforming image edges into a parameter space where circular shapes can be detected based on their radius and center position. The algorithm finds peaks in this space that correspond to potential circular objects.
- **Contour Area and Circularity Check:** Not all circular shapes detected are necessarily coins. Some may be reflections, overlapping objects, or other circular items. To filter out incorrect detections, we use contour analysis:
  - The area of the detected contour is compared to the expected area of a coin.

– The circularity of the object is determined using the formula:

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

where A is the area and P is the perimeter of the contour. A value close to 1 indicates a perfect circle, which helps filter out non-circular objects.

- **Size Filtering:** Coins typically fall within a known range of sizes. Objects that are significantly smaller or larger than expected are likely not coins and are removed. The algorithm uses a predefined size threshold based on the average radius of real-world coins.
- **Hierarchy Filtering:** To ensure individual coin detection in cases where coins are overlapping, we use contour hierarchy analysis. This prevents counting the same coin multiple times or misidentifying merged objects.

2.4 Results

The result of operations on `image-3` has been included in this report. Other results can be viewed in `Part-1/Output`.

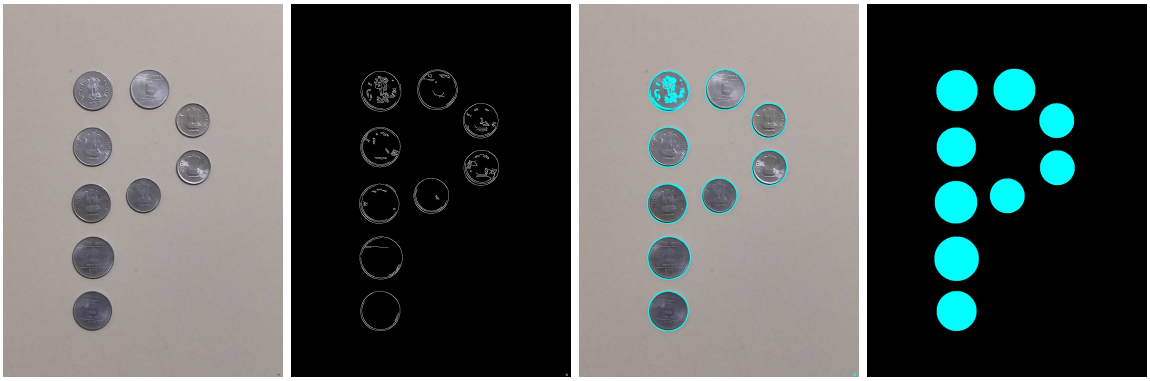

Figure 1: [Left-to-right] Original image, Image after applying canny edge detector, Detected coins, Segmented Coins

```
Processing image-3.jpeg...
-> Edge detection results stored in Output/image-3-results/edge-detection
-> Segmented coins saved in 'Output/image-3-results/isolated-coins'
-> Number of coins detected: 9
```

Figure 2: Output on terminal after execution

3 Image Stitching

3.1 Overview

The image stitching algorithm combines multiple images into a seamless panorama by detecting keypoints, matching features, aligning images, and blending them smoothly. The process ensures accurate alignment and minimal visual artifacts.

3.2 Methodology

The stitching process follows these steps:

1. Detect keypoints in images using the Scale-Invariant Feature Transform (SIFT) algorithm.
2. Visualize detected keypoints by drawing crosses at their locations.
3. Match keypoints between overlapping images using a brute-force matcher with L2 norm.
4. Filter unreliable matches using Lowe's ratio test for better accuracy.
5. Compute homography using RANSAC to align images.
6. Warp and blend images using distance-based weighting to reduce visible seams.
7. Crop black borders to refine the final stitched panorama.

3.3 Feature Detection and Matching

Feature detection is performed using **SIFT**, which extracts keypoints and descriptors that remain consistent despite changes in scale, rotation, and illumination. To enhance visualization, crosses are drawn at detected keypoints instead of the default circular markers.

Feature matching is carried out using a **brute-force matcher** with L2 norm, ensuring robust correspondence between keypoints in overlapping images. To further refine matches, **Lowe's ratio test** is applied, removing weak correspondences to improve alignment accuracy.

3.4 Homography and Image Alignment

Homography estimation is used to compute the transformation matrix that aligns images based on matched keypoints. The relationship is expressed as:

$$p' = Hp \tag{2}$$

where H represents the homography matrix, p is a point in one image, and p' is its transformed counterpart in another image. The RANSAC algorithm is applied to filter out outliers and compute a reliable homography. Once determined, images are warped using perspective transformation for alignment.

3.5 Blending and Finalization

To ensure a seamless panorama, overlapping regions are blended using a distance-based weighting technique. This method gradually transitions between images, reducing visible seams and improving the visual quality of the final output. As a final step, black borders caused by transformations are cropped, resulting in a refined and visually appealing stitched panorama.

3.6 Results

For each folder, three types of images are generated:

- Keypoints present within an image.
- Images matching the keypoints between contiguous blocks.
- Intermediate image after stitching 2 images together and so on, until a panorama is obtained.

Refer `Part-2/Output/Set-1` for all the produced images. For the sake of brevity, only a few images have been mentioned in this report.

Figure 3: Keypoints detected in `image-3` (left) and `image-4` (right) of `Set-1`

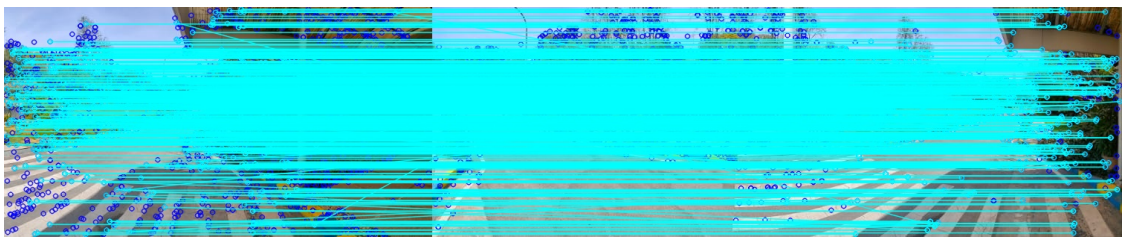

Figure 4: Keypoints matched between `image-3` and `image-4` of `Set-1`

Figure 5: Intermediate image obtained after stitching 3 blocks of images together

Figure 6: Final completely stitched panoramic image

4 Repository

https://github.com/PraveenPeterJay/VR_Assignment1_PraveenPeterJay_IMT2022064.git