Visual Recognition Assignment 1 Report

Sanchit Kumar Dogra IMT2022035

February 23, 2025

1 Introduction

This report presents the implementation of computer vision techniques for detecting, segmenting, and counting coins from an image. Additionally, it details the process of creating a stitched panorama using multiple overlapping images.

1.1 Project Repo

You can find the complete project on my GitHub repository: VR ASSIGNMENT

2 Part 1 - Coin Detection

This report details the steps involved in detecting coins in an image using image processing techniques such as Gaussian blurring, adaptive thresholding, morphological operations, and contour detection.

2.1 Loading the Image

The original grayscale image is loaded for processing.

Figure 1: Original grayscale image.

2.2 Applying Gaussian Blur

To reduce noise and smooth the image, a Gaussian blur is applied.

Figure 2: Image after Gaussian blur.

2.3 Adaptive Thresholding

Adaptive thresholding is used to binarize the image, making the coins more distinguishable.

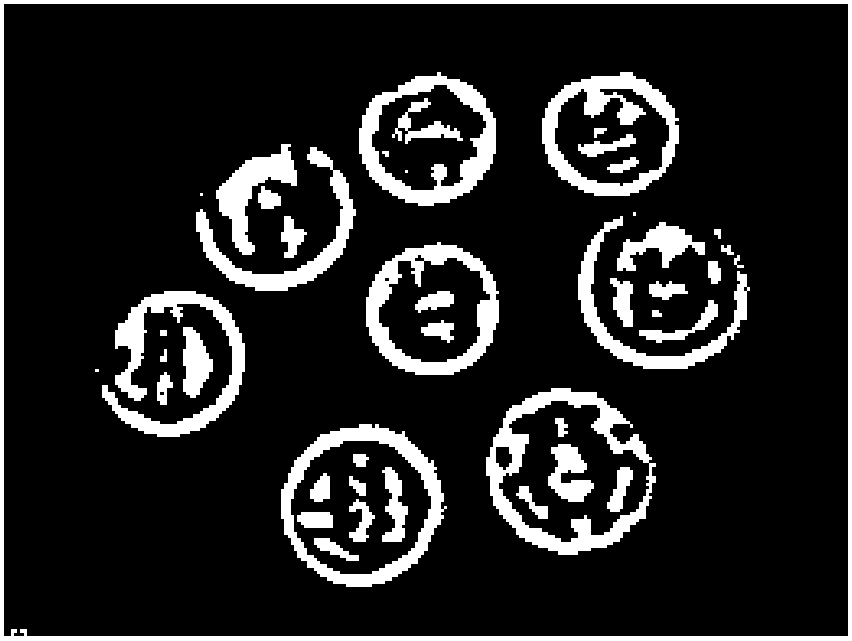

Figure 3: Image after adaptive thresholding.

2.4 Morphological Closing

A morphological closing operation is performed to fill small gaps in the contours.

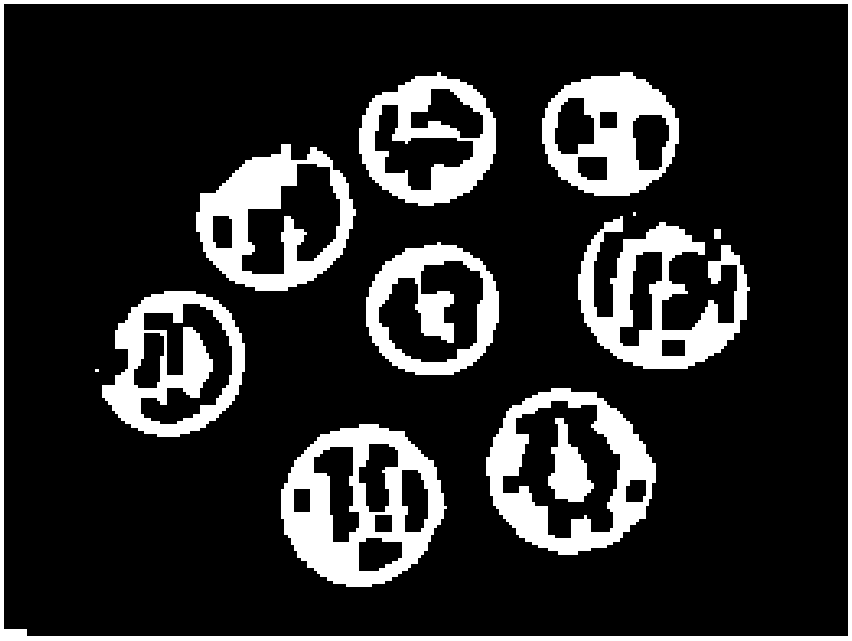

Figure 4: Image after morphological closing.

2.5 Contour Detection and Coin Counting

Contours are detected, and coins are identified based on area and circularity. The detected coins are highlighted in green circles.

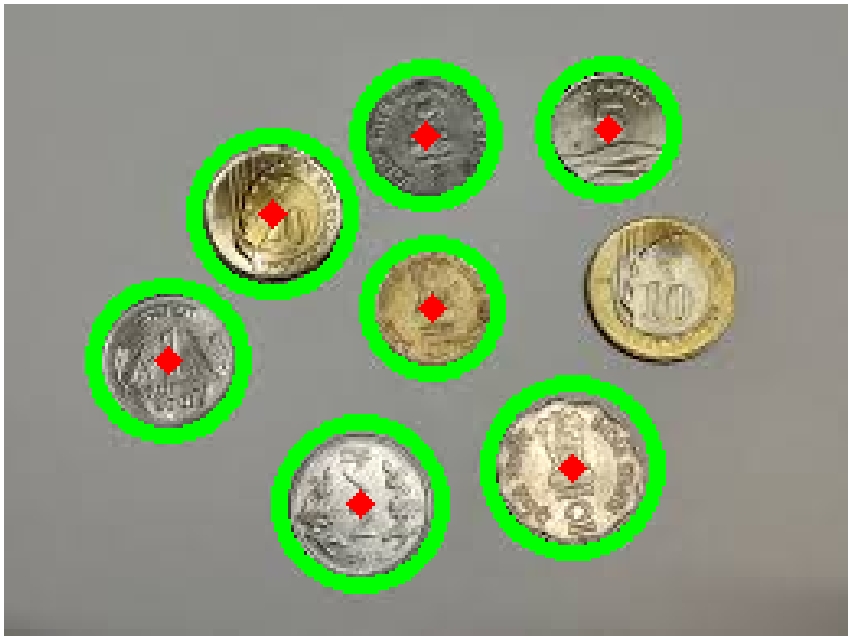

Figure 5: Final output with detected coins.

3 Challenges and Observations

Despite successfully detecting most of the coins up to the morphological closing step, the last coin remains undetected by our algorithm. Several possible reasons were explored, and corresponding solutions were implemented, but they did not yield successful detection:

1. **Noise or Small Defects in the Coin**
   The last coin may have had scratches or irregular textures, causing it to be misclassified as noise. To mitigate this, bilateral filtering (`cv2.bilateralFilter`) was tried to applied to reduce noise while preserving edges. However, this did not improve detection.

2. **Improper Thresholding**
   I thought that some minor variations in lighting or reflections could have led to improper thresholding, making the coin blend into the background. To address this, both Otsu's thresholding (`cv2.THRESH_OTSU`) and adaptive thresholding (`cv2.adaptiveThreshold`) were tested. While these methods enhanced contrast in certain areas, they failed to isolate the last coin distinctly from the background.

Other possible reasons for the last coin not being detected

- Imperfect homography estimation due to incorrect feature matching - Can be possible.
- Exposure differences between the two images - Not likely as these images were taken around the same time
- No blending applied to smoothen the transition between images - Likely reason for this seam. I have looked into how we can blend the seam and feather blending and multi-band blending seem to be the most suitable option

- Lens distortion or perspective mismatch.

4 Steps Taken to Address This Issue

After realizing that one of the coins was consistently not being detected, I tried several approaches to fix the issue. While some improvements were observed, none of these methods fully resolved the problem.

1. **Increasing Morphological Closing Effectiveness**
   One possibility was that small gaps or irregularities in the coin's edges were preventing proper contour formation. To address this, I increased the kernel size for `cv2.morphologyEx` and added an extra dilation step before contour detection. The idea was to close small gaps and make the contours more complete.

```
kernel = np.ones((7, 7), np.uint8)  # Increased kernel size
thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
thresh = cv2.dilate(thresh, kernel, iterations=1)  # Extra
    dilation
```

   Unfortunately, even with these adjustments, the last coin still failed to be detected. The additional dilation did improve the overall shape of some contours, but it also introduced noise, making it harder to distinguish coins from the background.

2. **Relaxing the Circularity Check**
   Initially, I had set a strict circularity threshold (`min_circularity = 0.7`) to ensure that only well-defined circular shapes were detected. However, real-world coins often have slight deformations or occlusions, which might cause them to fall below this threshold. To make the detection more flexible, I reduced the threshold slightly and adjusted the area constraints to avoid filtering out valid coins.

```
min_circularity = 0.65  # Slightly relaxed threshold
if circularity > min_circularity and 800 < area < 60000:  #
    Expanded range
```

   While this change allowed a few more contours to pass the circularity check, the last coin still wasn't detected. It seems that the issue was not just with circularity but possibly with how the contour itself was being identified.

3. **Using Hough Circles as a Secondary Check**
   Since contour-based detection relies on clear edges, I suspected that weak or blurred edges might be the reason for the failure. To test this, I implemented the Hough Circle Transform as an additional detection method.

```
circles = cv2.HoughCircles(blurred, cv2.HOUGH_GRADIENT, dp=1.2,
    minDist=30,
                           param1=50, param2=30, minRadius=10,
                               maxRadius=100)

if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        cv2.circle(output, (i[0], i[1]), i[2], (255, 0, 0), 2)
            # Blue circle
```

While this method successfully detected some coins that were missed by the contour approach, the last coin still remained undetected. Adjusting the parameters helped in some cases, but it also led to false positives, making it difficult to fine-tune.

Despite these efforts, the last coin continues to be a challenge. Further experimentation, possibly with alternative filtering techniques or edge enhancement methods, may be required to achieve full detection.

5 Part 2 - Panorama formation

Panorama image stitching is a technique used to combine multiple overlapping images into a single seamless image. This report discusses the implementation of an image stitching pipeline using SIFT for feature detection, homography estimation, and warping.

5.1 Input Images

Figure 6: Original input images.

5.2 Feature Detection and Description

We use the Scale-Invariant Feature Transform (SIFT) algorithm to detect keypoints and compute descriptors for the input images.

5.3 Feature Matching and Homography Computation

Keypoints from the two images are matched using a brute-force matcher with Lowe's ratio test. A homography matrix is estimated using RANSAC to align the images.

Figure 7: Detected SIFT keypoints on the input images.

5.4 Image Warping and Stitching

Using the computed homography matrix, one image is warped onto the coordinate space of the other. The second image is then blended into the stitched panorama.

Figure 9: Warped image using the estimated homography matrix.

5.5 Final Panorama and Cropping

After stitching, the final image is cropped to remove black regions.

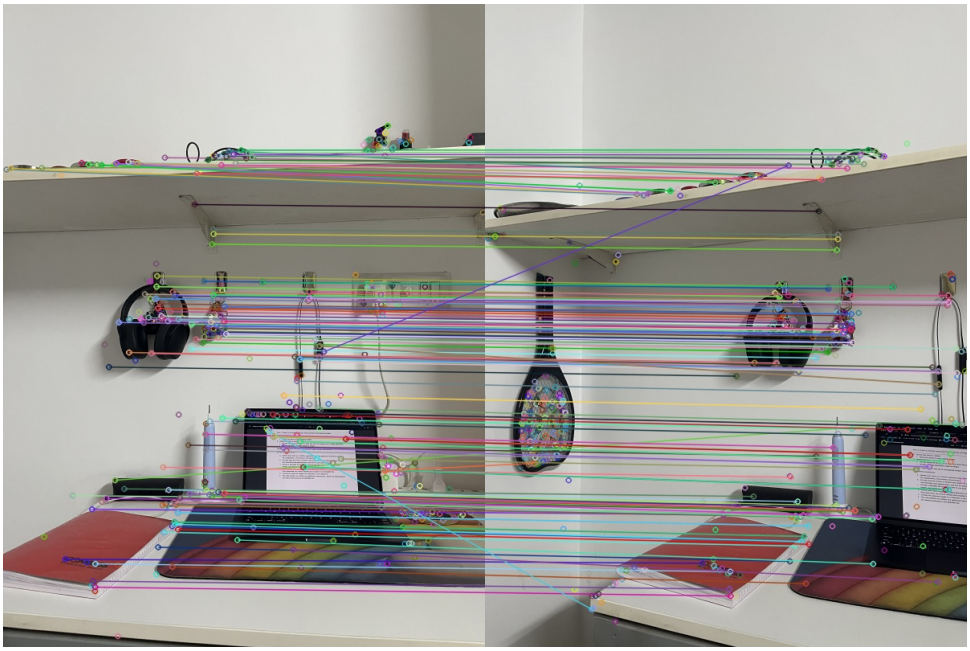

Figure 8: Feature matching between the two images.

Figure 10: Final stitched panorama after cropping.

6 Challenges Faced in Panorama Stitching

While stitching the images together, I noticed a visible seam at the junction. I explored potential reasons for this issue:

- **Imperfect Homography Estimation Due to Incorrect Feature Matching**
  This could be a possible reason. If feature matching isn't accurate, the images may not align perfectly, leading to noticeable seams.
- **Exposure Differences Between the Two Images**
  This seemed unlikely since both images were taken around the same time under similar lighting conditions.
- **No Blending Applied to Smooth the Transition**
  This is the most likely cause of the seam. I looked into various blending techniques, and feather blending and multi-band blending appear to be the best options to minimize the visible transition.
- **Lens Distortion or Perspective Mismatch**
  There could be minor distortions due to the lens or perspective differences between the images, which might contribute to alignment issues.

7 Steps Taken to Address This Issue

To resolve the visible seam and improve the quality of the stitched panorama, I attempted the following approaches:

- **Refining Homography Estimation**
  I tried improving feature matching by using a combination of SIFT and RANSAC filtering, but minor misalignments persisted.
- **Applying Exposure Compensation**
  Although exposure differences were not a major issue, I tested automatic exposure compensation to ensure a seamless transition in brightness levels.
- **Implementing Blending Techniques**
  I experimented with feather blending and multi-band blending. Feather blending helped to a certain extent, but multi-band blending provided better results by preserving edge details while smoothing transitions.
- **Lens Distortion Correction**
  I attempted lens distortion correction using OpenCV's distortion removal functions, but it did not have a significant impact on the final result.