VR-Assignment-1 (2/3/2025)

Jinesh Pagaria (IMT2022044)

Introduction

In this project, we aimed to detect edges, segment and count coins in an image in first part and in second we stitch images based on matching features. This report outlines the methods attempted, the results obtained, and the final approaches adopted.

Methods Attempted

<u>Coin detection, segmentation and counting</u>

1. Image Processing and Transformation

We applied several transformations to the input image to enhance visual clarity and prepare it for further analysis:

Grayscale Conversion: Simplified the image by reducing color channels to a single intensity channel.
Image Negative: Inverted pixel intensities to highlight contrast and reveal hidden details.
Contrast Stretching: Enhanced overall contrast by distributing pixel intensities across the entire possible range.
Histogram Equalization: Adjusted the contrast based on the image's histogram, leading to a more uniform intensity distribution.

2. Edge Detection and Highlighting

We used the Canny edge detector to identify object boundaries:

Applied Gaussian blurring to reduce noise and smoothen the image.
Performed edge detection using the Canny algorithm.
Highlighted the detected edges in green for better visualization.

What Worked:

Gaussian blurring effectively reduced noise without significant loss of detail.
Canny edge detection highlighted object boundaries with precision.

What Didn't:

Without sufficient blurring, the edge detection was noisy and captured irrelevant details. Image negative gave false edges in some cases.

Final Approach: A balanced Gaussian blur followed by Canny edge detection provided clear and accurate edge maps. Regardless of the transformation used, Canny was able to detect edges more or less.

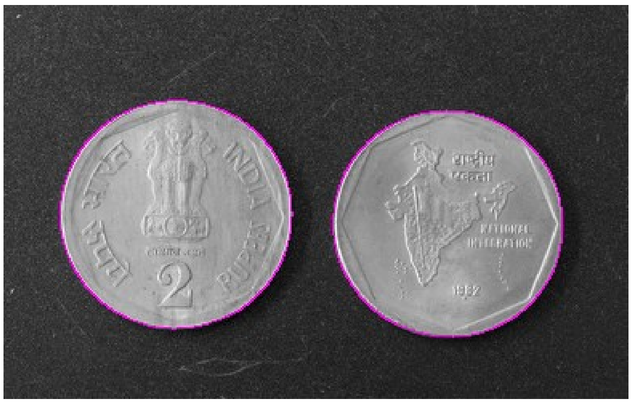


IMAGE 1

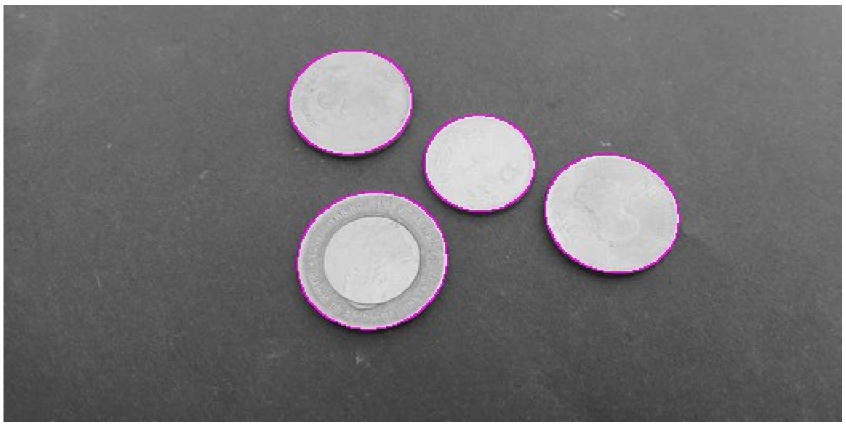


IMAGE 2

(HIGHLIGHTED EDGES IN GRAY-SCALE)

3. Region-Based Segmentation and Contour Detection

We segmented individual objects (coins) from the image:

Applied Gaussian blurring with a larger kernel for better object separation.
Converted the image to binary form using thresholding.
Detected object contours and drew bounding boxes around each.

What Worked:

Binary thresholding clearly separated objects from the background.
Contour detection accurately identified and outlined individual objects.

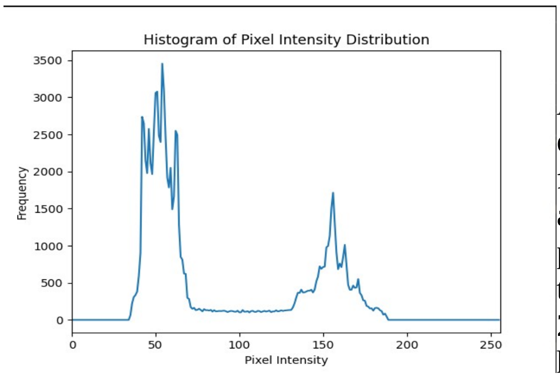
What Didn't:

Over-blurring sometimes merged closely positioned objects. amount of the area encapsulated by contour also matters as size of coins might be different in each.

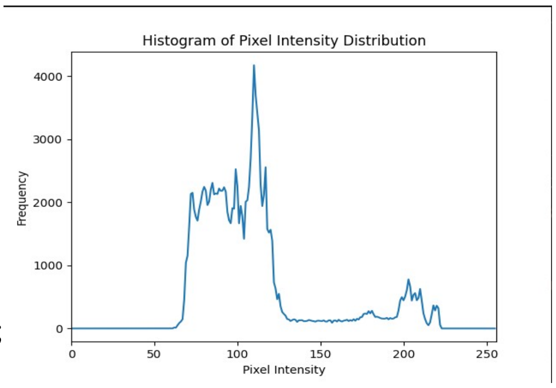
Final Approach: Moderate blurring combined with binary thresholding and contour detection provided the most accurate segmentation. Amount of area encapsulated by contour matters and must be set accordingly.

For binary thresholding we plot histogram distribution of intensities and final the 'knee' intensity.

This plot is done by /scripts/help.py. For the 2 images shown above the histograms are in order left to right respectively:



As is evident from above plots, there are 2 clusters; higher



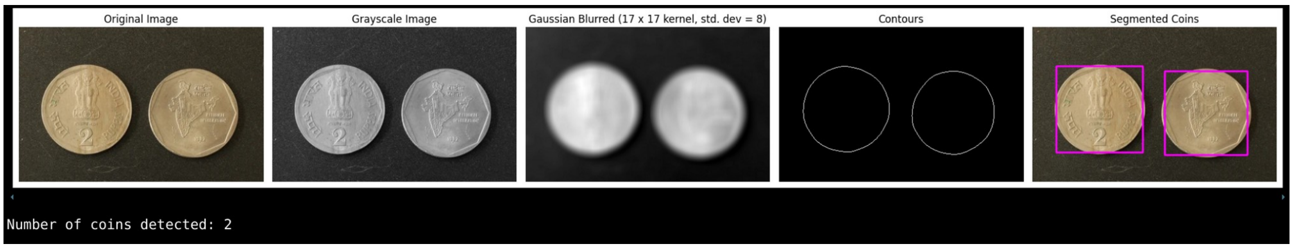
one corresponding to coins. So in /scripts/1.ipynb we set x = 130 for first image and x = 160 for the second. Line:

```
_, thresh = cv2.threshold(blurred, 160, 255, cv2.THRESH_BINARY)
```

The above line puts 160 in place of the parameter x and output obtained is for image 2.

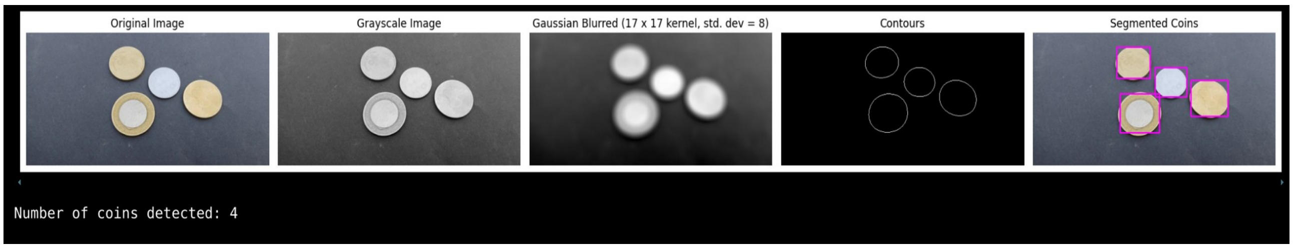
```
_, thresh = cv2.threshold(blurred, 130, 255, cv2.THRESH_BINARY)
```

The above line puts 130 in place of the parameter x and output obtained is for image 1.



```
Number of coins detected: 2
```

For image 1



```
Number of coins detected: 4
```

For image 2

The above 2 images are taken just as an example for this process. Both images saved as /input/img1.png and /input/img2.png

Feature Detection and Image Stitching

We used the SIFT algorithm for keypoint extraction and image alignment:

- Detected distinctive keypoints and computed descriptors.
- Matched keypoints between images using the BFMatcher.
- Estimated homography for geometric transformation.
- Stitched images into a seamless panorama.

What Worked:

- SIFT provided robust and scale-invariant feature detection.
- Homography estimation aligned images accurately.

What Didn't:

- Keypoint matching failed when images had insufficient overlap.

Final Approach: SIFT with careful parameter tuning and homography estimation produced a well-aligned stitched image.

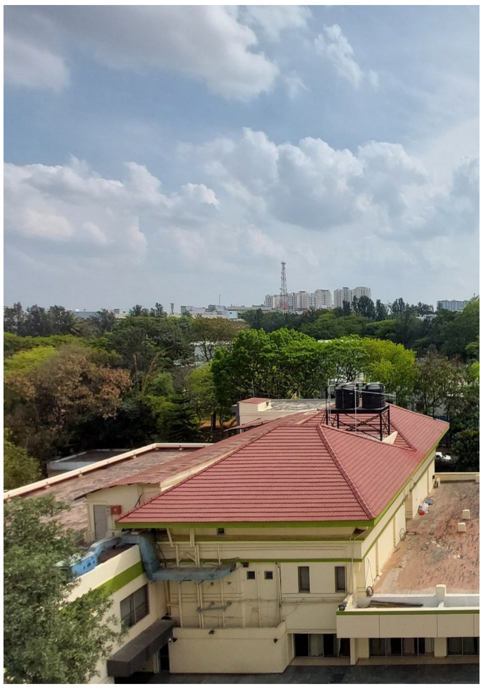


IMAGE 1

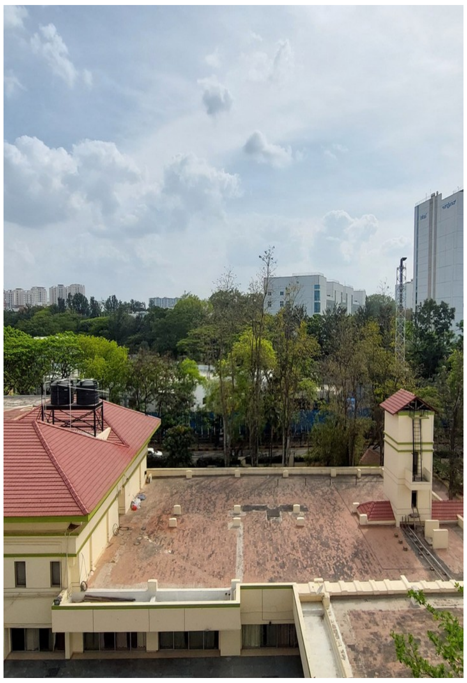
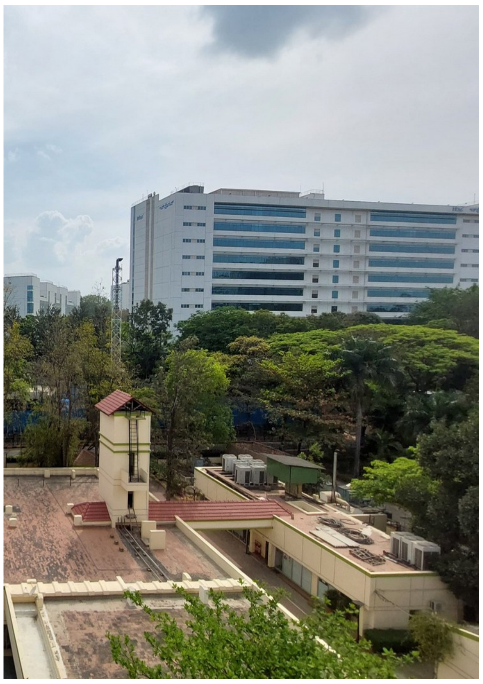
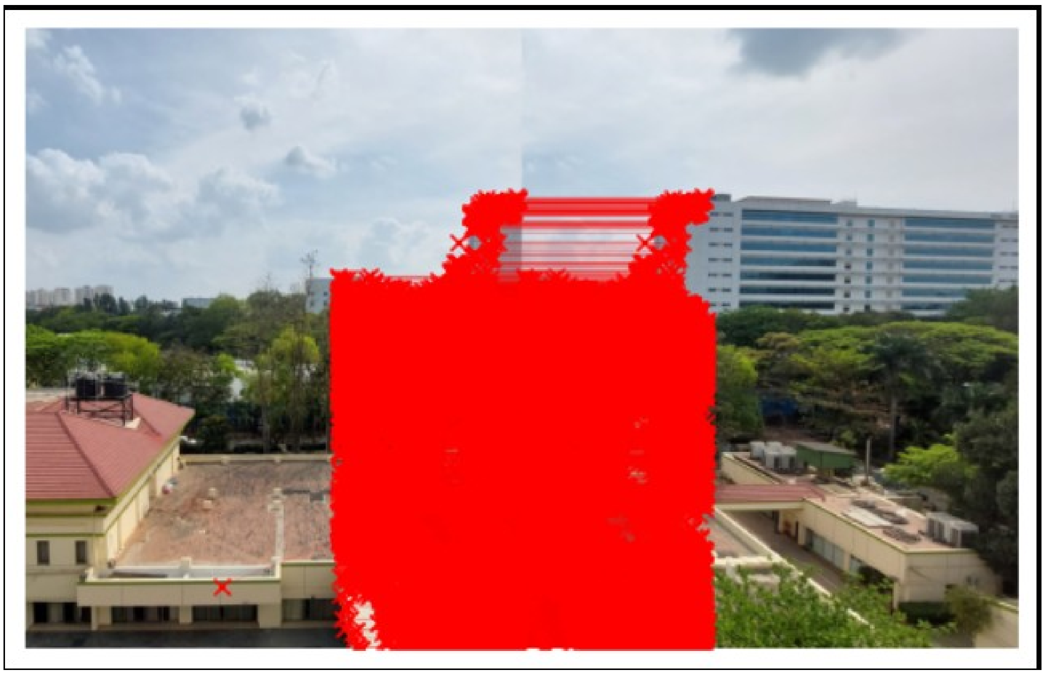


IMAGE 2

IMAGE 3





(MATCHING POINTS FOR IMAGE 2 AND IMAGE 3)

(MATCHING POINTS FOR IMAGE 1 AND STITCHED IMAGE OF IMAGE 2 AND IMAGE 3)