AIM 825- Sec-B: Visual Recognition
Assignment-1

Keshav Goyal
IMT2022560
Keshav.Goyal560@iiitb.ac.in

1

1. Introduction

The goal of this assignment is to analyze and process images using computer vision techniques for two tasks: coin detection and segmentation, and image stitching. In the first part, edge detection is applied to identify coins, followed by region-based segmentation to isolate individual coins, and a counting function to determine the total number of coins in the image. In the second part, key points are extracted from overlapping images to align and stitch them into a seamless panorama.

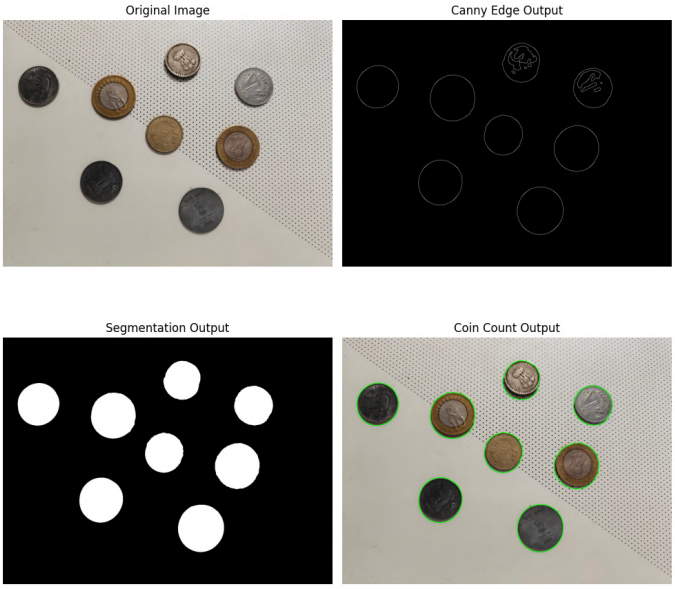
2. Question 1

2.1. Code workflow Description:

- **Loading the data:** A total of five images were sourced from the web and captured manually to detect and count coins.
- **Preprocessing the image:**
  - The image was first converted to grayscale and then blurred using a Gaussian blur with a 15×15 kernel and a standard deviation of 7 to reduce noise.
  - Next, the grayscale image was converted into a binary (black and white) image. The **THRESH_OTSU** method automatically determined the optimal threshold value, while **THRESH_BINARY_INV** ensured that the background appeared black and the objects of interest (coins) appeared white.
  - To remove noise and fill gaps in the objects, **Morphological Closing** was applied using two iterations:
    - *Dilating:* Expands the white regions by growing object boundaries.
    - *Erosion:* Shrinks the white regions back to their original size, refining the segmentation.
- **Edge detection:** Canny Edge detector was used for edge detection, where only edges between 120 and 250 are used as strong edges.
- **Image Segmentation:** Using the binary image, we find a list of detected contours(ignoring the outermost contours). We create a dummy black image which is of the same size as grayscale image, and that image will be used to draw and fill the detected contours.
- **Counting Coins:** Contours are created from the image from Canny Edge detector and only countours with area > *475* are taken removing the small contours and they are drawn on an image.

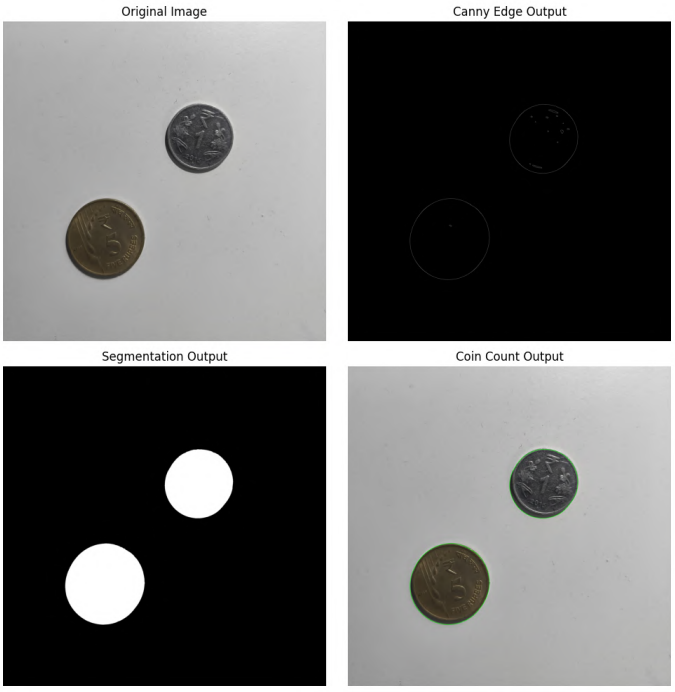
2.2. Results

- **Image 1**



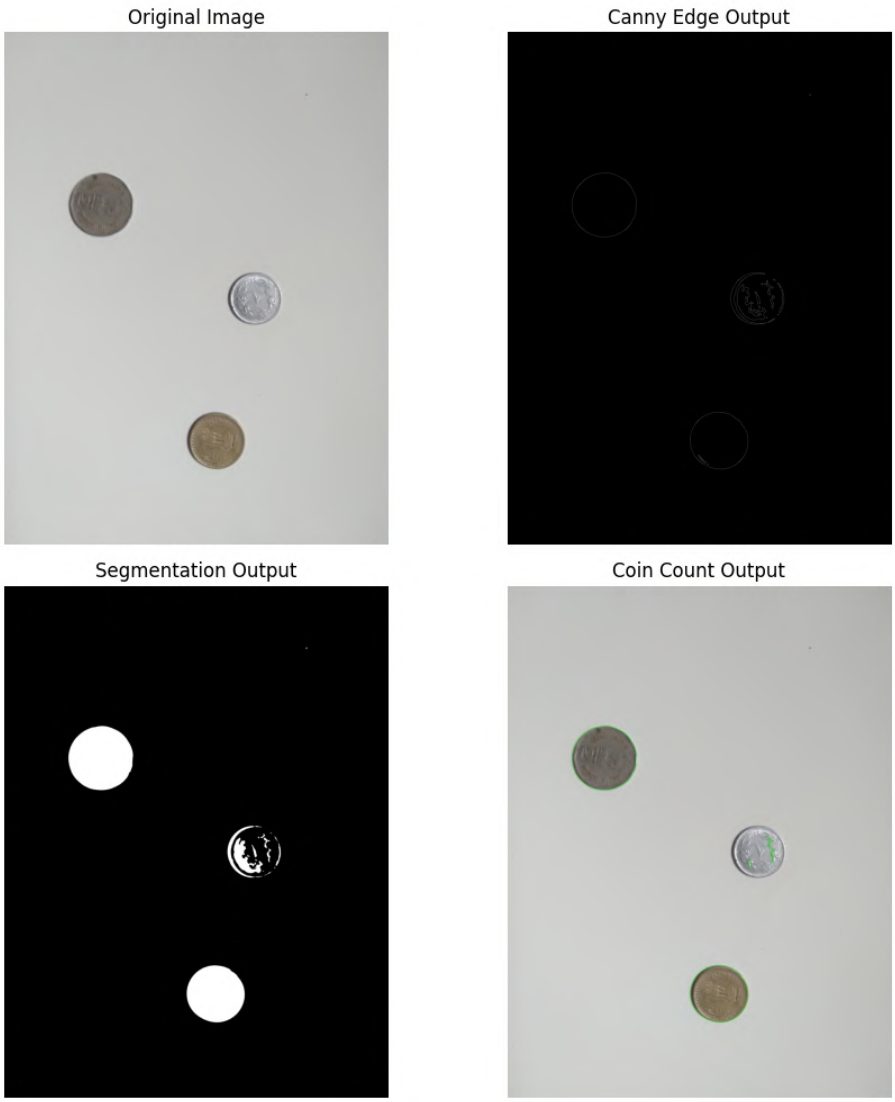
The number of coins detected for this image is correct: 8

- **Image 2**



The number of coins detected for this image is correct: 2

- **Image 3**



The number of coins detected for this image is incorrect: 5

The coin detection algorithm fails in certain cases, like image 3, due to poor contrast between coins and the background, making thresholding ineffective. Uneven lighting and shadows cause intensity variations, leading to misclassification of coins. Noise in the image creates false contours, affecting segmentation, while overlapping coins are sometimes detected as a single object due to the lack of a separation mechanism like watershed segmentation. Additionally, the contour filtering step may ignore smaller coins if the area threshold is too high or count noise if it's too low.

3. Question 2

3.1. Code workflow Description:

- **Loading the images:** Three images were captured using my phone camera for image stitching. Below are the three images taken:

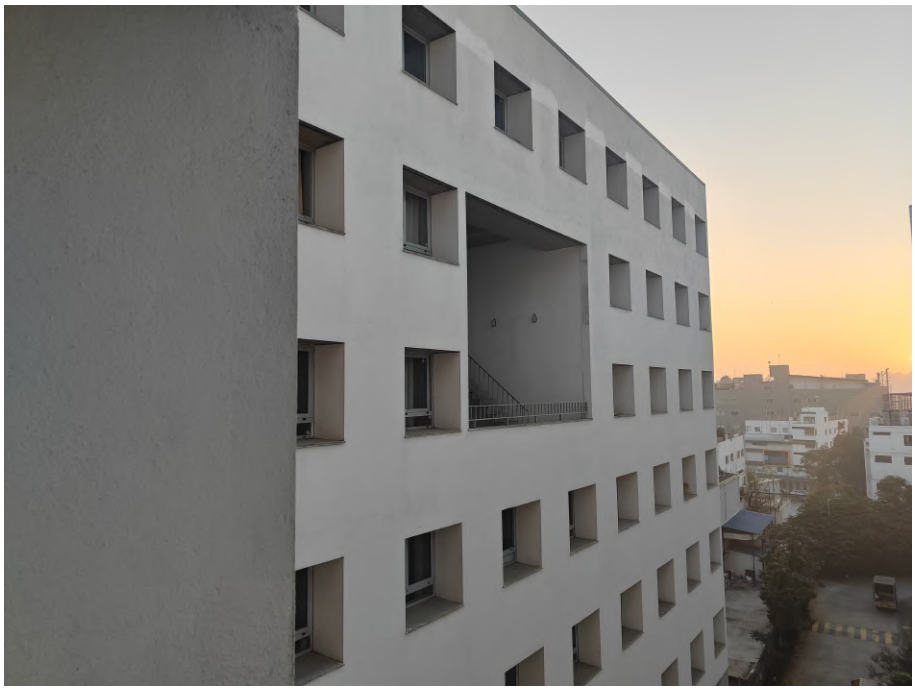


Image 1

Image 2

Image 3

- **Feature Detection: ORB: Oriented Fast and Rotated BRIEF** was used to detect keypoints in the image. Here are all the keypoints in red.

Image 1 with keypoints

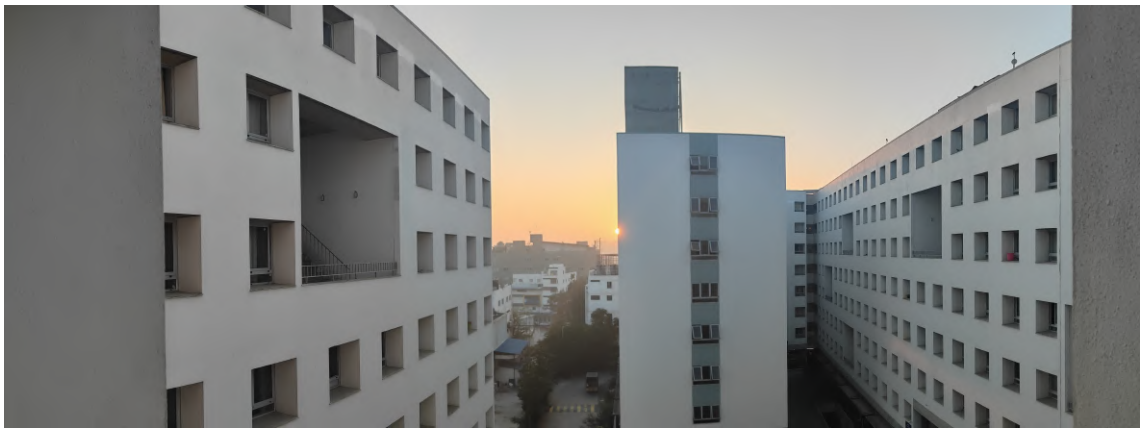
Image 2 with keypoints

Image 3 with keypoints

- **Image Stitching:** Now we use OpenCV's Stitcher class to merge the images into a single stitched image. The image formed using this looks like:

Stitched Image with messed up borders

To correct the distorted borders, a black border of width 5 was added around the stitched image. The image was then converted to grayscale and thresholding was applied to separate the main stitched region from the background.
Next, contours were detected in the thresholded image, and the largest contour—corresponding to the main stitched area—was selected. A binary mask of the stitched region was created, and **morphological erosion** was applied to refine the mask and remove extra background. Finally, the cropped version of the stitched image was extracted, as shown below:



Final Stitched Image

4. Github Link

Here is the github link for the assignment: VR_Assignment1_Keshav_IMT2022560