Visual Recognition Assignment –1

Narayana Srimanth Udayagiri-IMT2022052

Git hub repo: https://github.com/narayana-srimanth/VR_Assignment1_Narayana_IMT2022052

PART-1 Coin Detection:

1. Introduction

This report presents a coin detection system using image processing techniques in OpenCV and this approach involves grayscale conversion, thresholding, morphological operations, and contour detection to identify coins in an image. The system is implemented in Python and outputs segmented and detected coin images.

2. Methodology

2.1 Image Preprocessing

- The input image is read using OpenCV and converted from BGR to grayscale.
- To reduce noise, median blurring is applied with a kernel size of 7.

- Thresholding is performed using Otsu's method to obtain a binary image.
- Morphological closing is applied to remove small noise and enhance segmentation.

2.2 Segmentation and Contour Detection

- The processed binary image is labeled using the `measure.label` function from skimage to segment different connected components.
- The `color. label2rgb` function is used to visualize the segmented regions.
- External contours are extracted using `cv2.findContours`.
- The detected contours are drawn on the original image for visualization.

2.3 Output and Visualization

- The detected coin image and segmented coin image are saved in the output directory.
- Matplotlib is used to display the original, segmented, and detected images.

3. Experimentation

During experimentation, two different blurring techniques were tested:

1. **Gaussian Blurring:** It was observed that Gaussian blurring introduced noise inside the coin regions, making detection less effective.

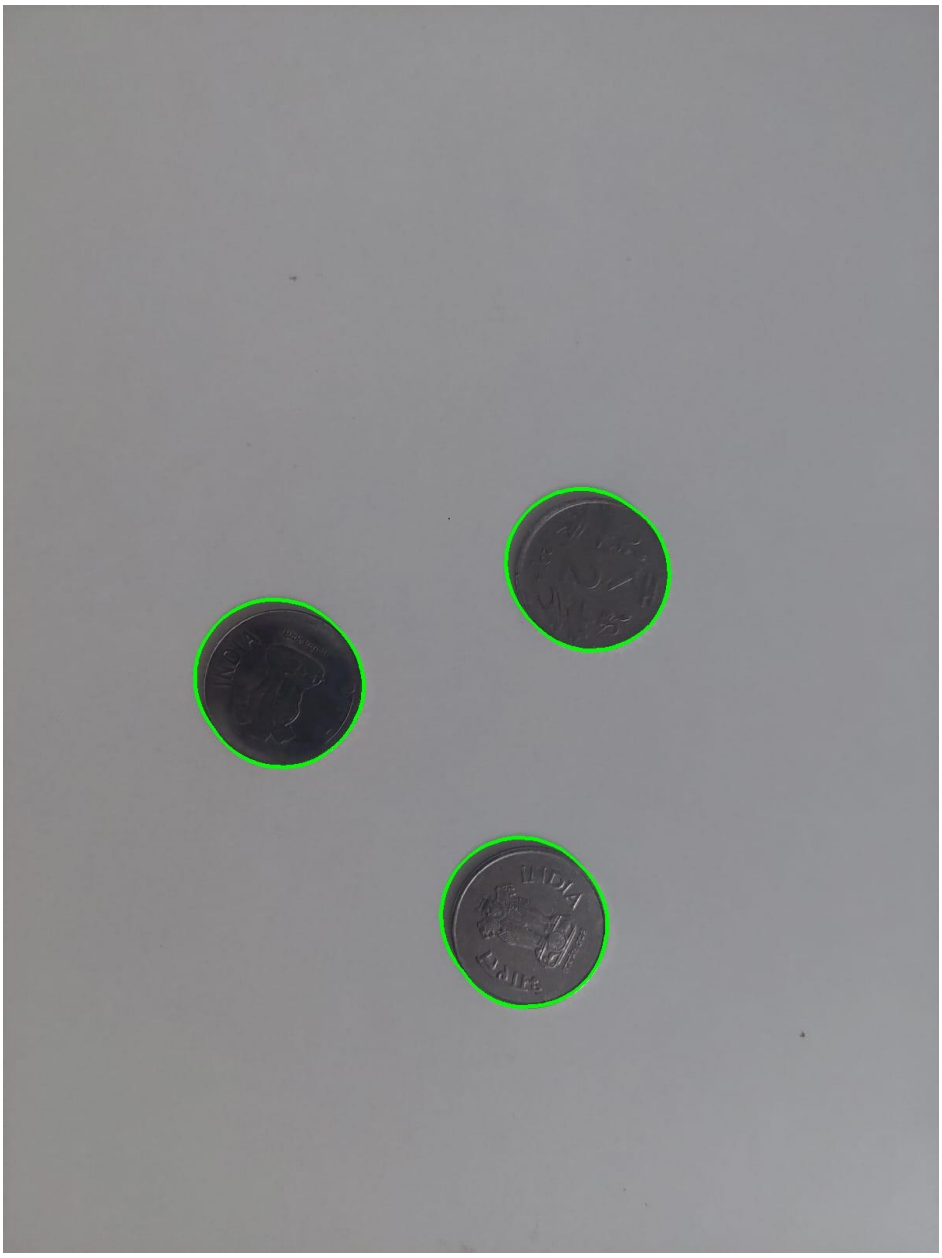
2. **Median Blurring:** In contrast, median blurring effectively removed noise without affecting the coin boundaries, leading to more accurate detection.

Based on these observations, median blurring was chosen as the optimal technique for preprocessing.

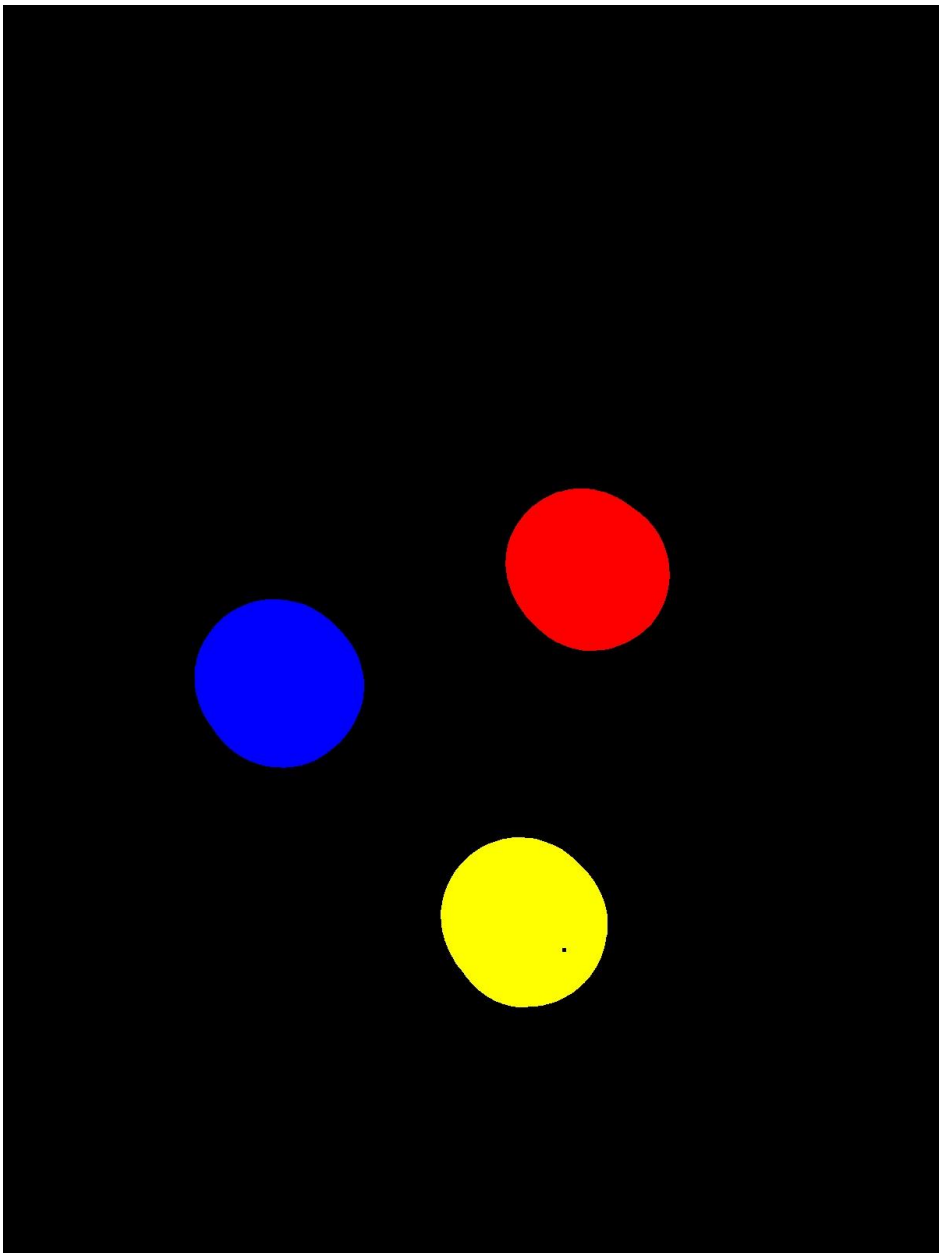
4. Results and Conclusion

Input Image:

Detected Coins:



Segmented Coins:



- The system successfully detects and counts the number of coins in the given image.
- The segmentation approach, using labeling and contour detection, provides a reliable method for coin detection.
- The final system is capable of processing coin images and saving the results for further analysis.

PART-2 Image Stitching:

1. Introduction

In this section we will see an image stitching system that combines two images using feature-based techniques. The approach involves keypoint detection, feature matching, homography computation, warping, and blending to generate a seamless panorama. The system is implemented in Python using OpenCV.

2. Methodology

2.1 Image Preprocessing

- The input images are loaded and converted from BGR to RGB and grayscale formats for feature extraction.

2.2 Feature Detection and Matching

- The SIFT (Scale-Invariant Feature Transform) algorithm is used to detect keypoints and compute descriptors.
- A brute-force matcher with L2 norm and cross-checking is applied to find the best feature matches.
- The best matches are sorted based on feature distance and visualized.

2.3 Homography Computation

- The RANSAC algorithm estimates a homography matrix using the best feature matches.

- The homography matrix helps in aligning one image with the other.

2.4 Image Warping and Blending

- One image is warped using the homography matrix to align with the second image.
- The warped image is blended with the second image to create a stitched panorama.

2.5 Black Border Removal

- The stitched image is processed to remove black regions using binary thresholding and bounding box cropping.

3. Experimentation

During experimentation, different feature detection and matching approaches were explored:

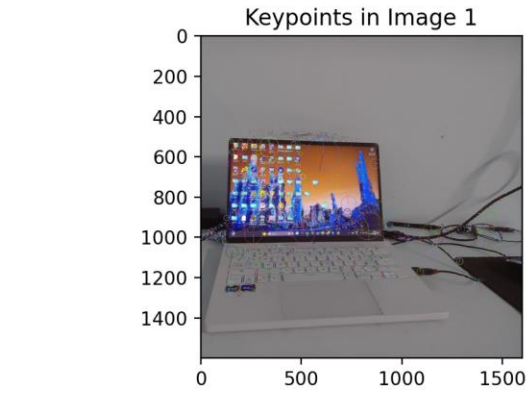
1. **SIFT with BFMatcher:** Provided reliable feature matching, leading to a well-aligned panorama.
2. **Removing Black Borders:** Cropping black areas improved the final output's appearance.

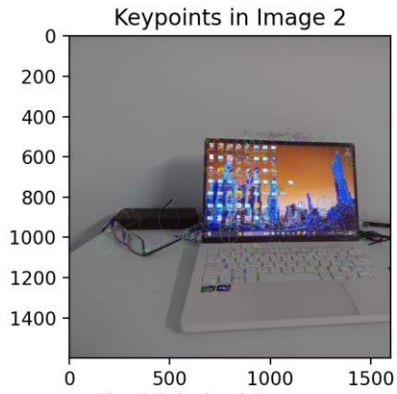
Based on these observations, SIFT with BFMatcher was chosen for robust image alignment.

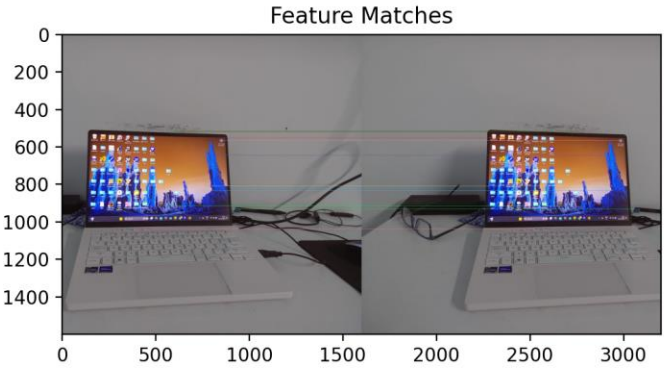
4. Results and Conclusion

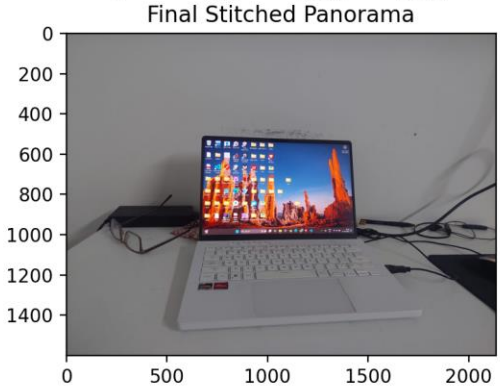
Input Image:

Output Imge:









- The system successfully stitches two images into a panorama.
- SIFT provided accurate feature detection and matching.
- Homography estimation and warping ensured proper image alignment.
- The black region removal step improved the final panorama quality.