Coin Detection and Image Stitching Project Report

H.Sanjeev-IMT2022530 GITHUB LINK:
https://github.com/Sanjeevharge/IMT2022530_vr_assignment11/tree/master

---

1. Introduction

This project is a combination of two essential computer vision tasks:

1. Coin Detection, Segmentation, and Counting - This module employs the Hough Circle Transform to detect circular objects in an image, followed by segmentation and counting.
2. Image Stitching - This module uses feature-based image alignment techniques to merge overlapping images into a seamless panorama.

The project leverages OpenCV-based image processing techniques to extract meaningful information from images and automate these tasks effectively.

---

2. Part 1: Coin Detection, Segmentation, and Counting

2.1 Overview

This section deals with identifying circular objects (coins) within an image, segmenting them individually, and counting the total number of detected coins. The approach involves advanced edge detection and shape analysis using the Hough Circle Transform, followed by precise image segmentation.

2.2 Steps Involved

Step 1: Preprocessing & Edge Detection

- Convert the input image to grayscale to reduce complexity and enhance processing speed.
- Apply Gaussian Blur to minimize noise and improve edge detection accuracy.
- Utilize Canny Edge Detection to enhance object boundaries and facilitate circle detection.

Step 2: Circle Detection using Hough Transform

- Implement the Hough Circle Transform to analyze pixel gradients and detect circular shapes within the image.
- Identify and highlight detected circles by overlaying them onto the original image for clear visualization.

Step 3: Segmentation & Counting

- Extract individual detected coins by cropping regions based on detected circle coordinates.
- Save each segmented coin as a separate image file for additional processing.
- Display the total number of detected coins in the terminal output.

2.3 Methods Used

- **Hough Circle Transform: A robust technique to detect circular shapes in images.**
- **Gaussian Blur: A filtering technique to reduce image noise and enhance detection accuracy.**
- **Canny Edge Detection: A widely used algorithm for detecting object boundaries.**
- **Image Cropping: Extracts individual coin regions based on detected coordinates.**

2.4 Implementation

Requirements

To execute this module, install the required dependencies using:

```
pip install opencv-python numpy pandas matplotlib
```

How to Run

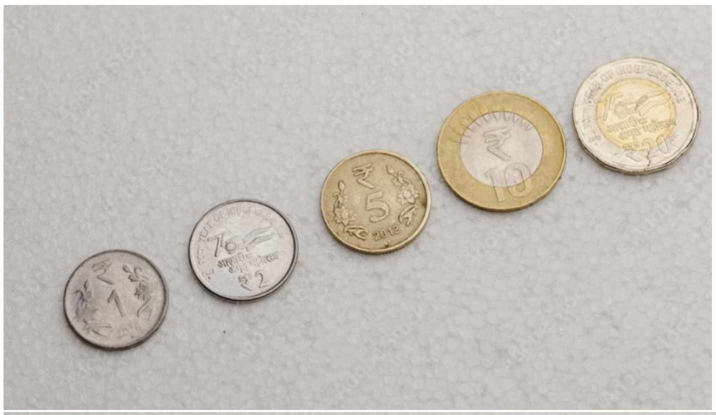
1. **Place the input image (coins2.png) in the project directory.**
2. **Run the script:**

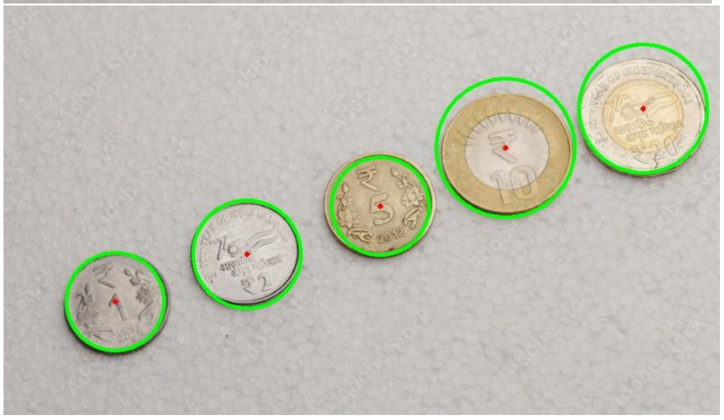
```
python coin_detection.py
```

2.5 Example Outputs

- **Input Image: The original image containing multiple coins.**
- **Detected Coins Image: An annotated image with circles drawn around detected coins.**
- **Segmented Coins: Cropped images of individual coins saved separately.**

INPUT and OUTPUT:

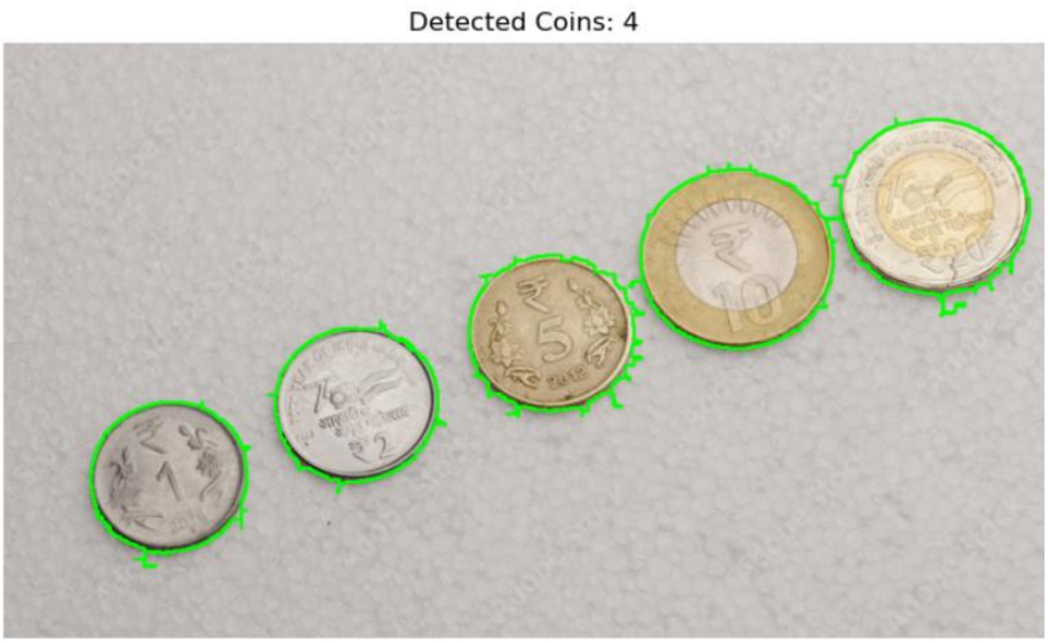




Previous method and results:

2.6 First Approach -Contour-Based Detection

- **Uses adaptive thresholding and morphological operations to preprocess the image.**
- **Applies Canny edge detection to enhance edges.**
- **But this approach couldn't divide up the 5 coins but detected only 4, as it took the last 2 coins as 1 unit**



3. Part 2: Image Stitching

3.1 Overview

This section deals with merging two overlapping images into a seamless panorama by detecting and matching key features between them.

3.2 Steps Involved

Step 1: Feature Detection & Matching

- **Convert both input images to grayscale for enhanced feature extraction.**
- **Extract key feature points using the Scale-Invariant Feature Transform (SIFT) algorithm.**
- **Use the FLANN (Fast Library for Approximate Nearest Neighbors) matcher to match keypoints between images.**
- **Filter and refine matches using Lowe's ratio test to retain only the most accurate correspondences.**

Step 2: Homography Estimation & Image Warping

- **Compute the homography matrix using RANSAC (Random Sample Consensus) to determine the transformation between images.**
- **Apply perspective warping to align one image with the other using the computed homography.**
- **Merge the transformed images seamlessly to generate a panoramic output.**

3.3 Methods Used

- **SIFT (Scale-Invariant Feature Transform): Extracts distinctive keypoints from images.**
- **FLANN Matcher: Efficiently finds matching keypoints across images.**
- **Homography Estimation: Establishes a geometric transformation between images.**
- **Perspective Warping: Warps one image to align with another seamlessly.**

3.4 Implementation

Requirements

To execute this module, install the required dependencies using:

```
pip install opencv-python numpy matplotlib
```

How to Run

1. **Place the overlapping images (p.jpeg and q.jpeg) in the project directory.**
2. **Run the script:**

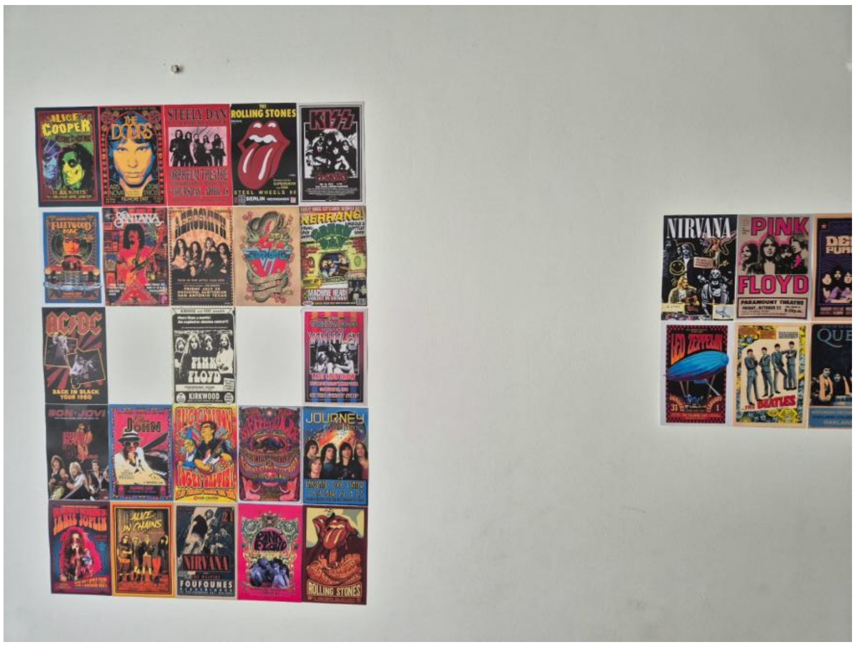
```
python image_stitching.py
```

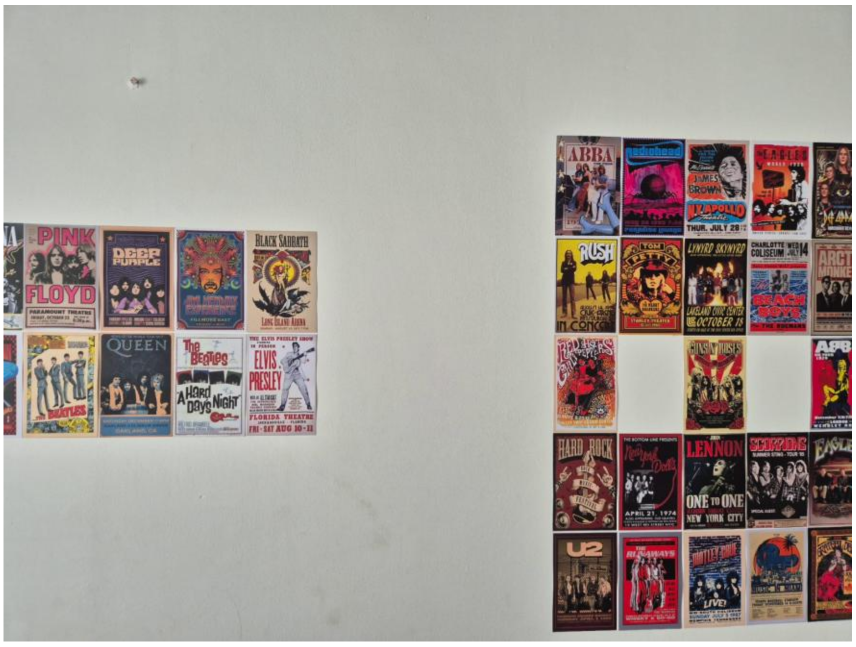
3.5 Example Outputs

- **Input Images: The two images before stitching.**

- **Detected Keypoints: Features identified in both images.**
- **Matched Keypoints: Correspondences between keypoints in both images.**
- **Final Stitched Panorama: The seamlessly combined panoramic image.**

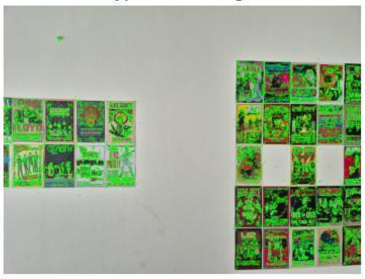
INPUT:



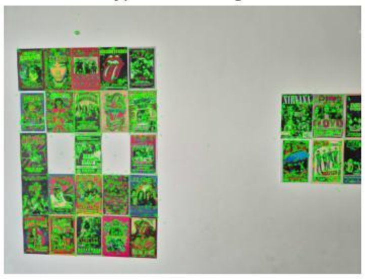


OUTPUT:

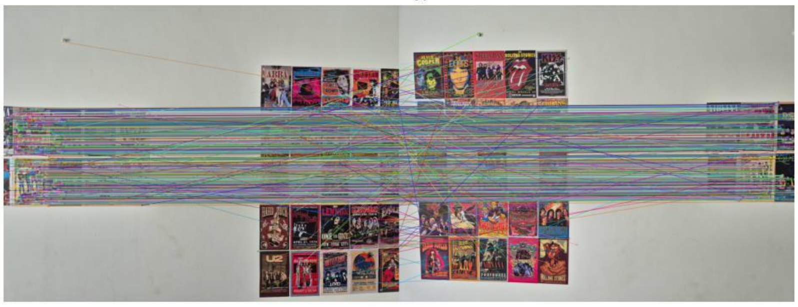
Keypoints in Image 1

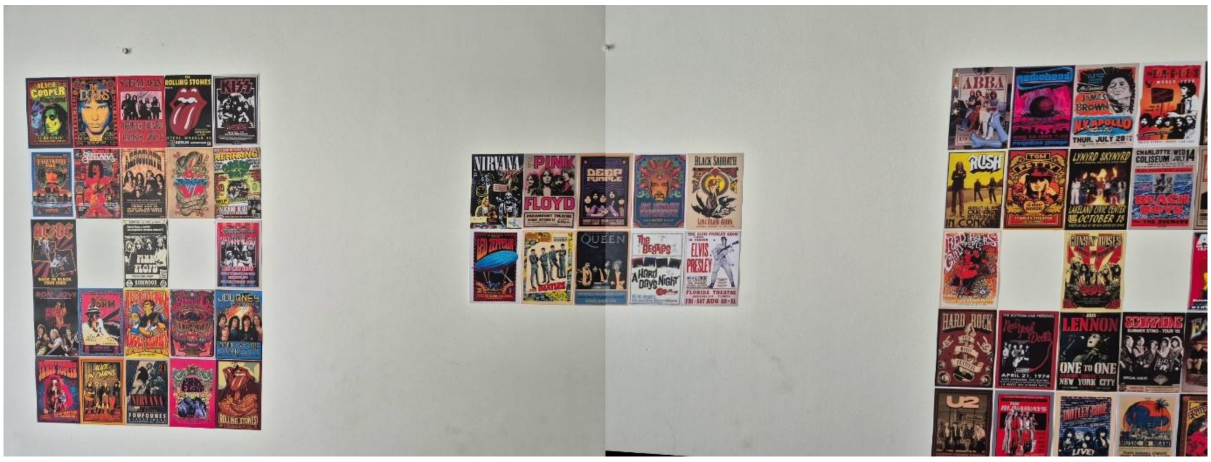


Keypoints in Image 2



Matched Keypoints

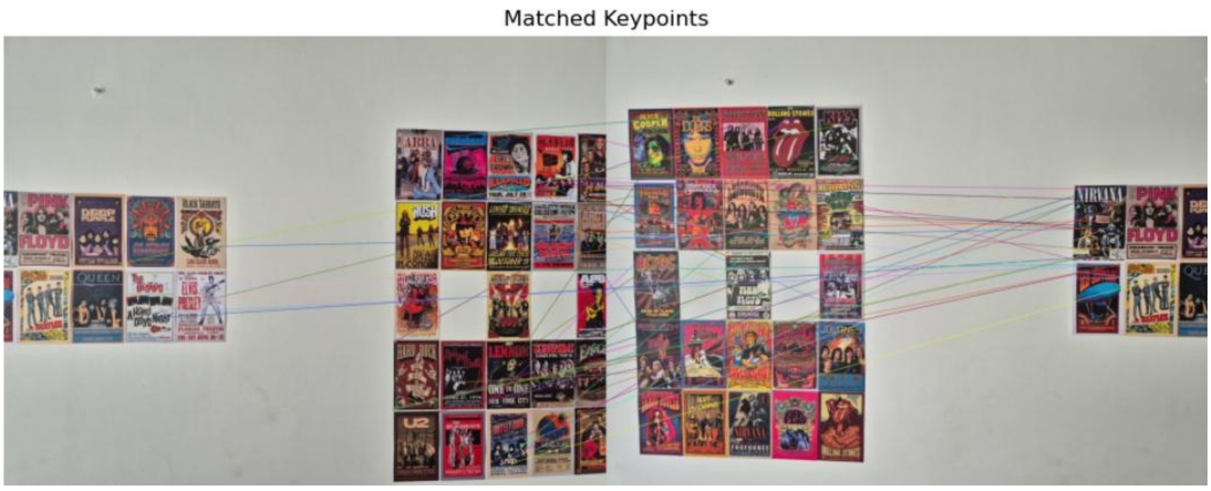


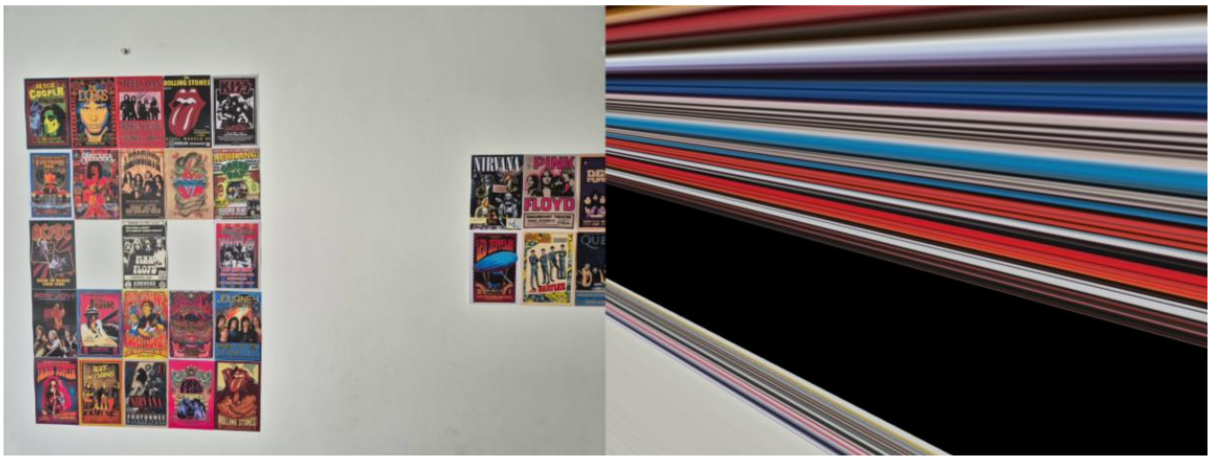


3.6 Previous methods:

Aspect	SIFT+FLANN(final code)	ORB+BFMatcher(initial)
Feature detector	SIFT	ORB
Description type	Floating point	Binary
Matching method	FLANN	BFMatcher
Matching criteria	Lowe's ratio	Distance sorting(top 50)

Output from the initial code:





4. Folder Structure

```
project_folder/
|— coin_detection.py  (Coin Detection & Segmentation)
|— image_stitching.py  (Panorama Stitching)
|— input_images/
|    ├— coins2.png
|    ├— p.jpeg
|    ├— q.jpeg
|— output_images/
|    ├— detected_coins.png
|    ├— coin_1.png
|    ├— keypoints1.jpg
|    ├— keypoints2.jpg
|    ├— matched_keypoints.jpg
|    ├— stitched_panorama.jpg
|— README.md
```

---

5. Results & Observations

5.1 Coin Detection & Segmentation

- **The system accurately detects circular coins using the Hough Circle Transform.**
- **Successfully segments individual coins by cropping their detected boundaries.**
- **Outputs the final count of detected coins in the terminal, ensuring correct quantification.**

5.2 Image Stitching

- **SIFT keypoints are accurately extracted, ensuring reliable feature detection.**
- **The FLANN matcher effectively finds corresponding keypoints between images.**
- **The final panorama is generated seamlessly using homography and RANSAC, producing minimal distortion and an aesthetically pleasing result.**

---

6. Conclusion

This project successfully implements two fundamental computer vision tasks:

- **Coin detection and segmentation effectively identify and count circular objects in an image.**
- **Image stitching leverages feature-based image alignment techniques to create seamless panoramas.**

These techniques have vast applications in areas such as automated object detection, surveillance, and panoramic image generation.

---

7. Future Improvements

- **Enhance coin detection accuracy using deep learning-based object detection models such as YOLO or Faster R-CNN.**
- **Improve image stitching by enabling multi-image panorama creation with advanced blending techniques.**
- **Optimize computational efficiency by utilizing GPU acceleration for faster feature extraction and matching.**
