# VR_Assignment1_ShashankVyas_MT2024141

**Title:** Coin Detection, Segmentation, and Panorama Creation

# 1. Introduction

In this assignment, the two computer vision tasks are:

1. **Coin Detection & Segmentation:**
   Detecting coins in an image, segmenting them individually, and counting the total number of coins.

2. **Panorama Creation:**
   Stitching multiple images into a single panoramic image using feature extraction, feature matching, homography estimation, and image warping.

---

# 2. Problem Statement

1. **Coin Detection & Segmentation**

    - **Objective:** Given an image containing multiple coins, identify each coin, segment them from the background, and count how many coins are present in the image.
2. **Panorama Creation**

    - **Objective:** Given multiple overlapping images of the same scene, create a seamless panorama by aligning and stitching them together.

---

# 3. Methods and Algorithms

## 3.1 Coin Detection & Segmentation

1. **Preprocessing**

    - **Grayscale Conversion:** The input image is converted from BGR to grayscale to simplify processing.
    - **Gaussian Blurring:** A Gaussian filter is applied to reduce noise and smooth the image.
2. **Edge Detection**

    - **Canny Edge Detection:** Used to detect edges by computing intensity gradients. This step helps in finding the boundaries of coins.
3. **Contour Detection**
    - **Contour Extraction:** OpenCV's `findContours` function retrieves the outlines of objects (coins) from the edge map.
4. **Segmentation**
    - **Area-Based Filtering:** Each contour is evaluated based on its area to filter out irrelevant objects (too small or too large).
    - **Bounding Box Extraction:** Valid contours are enclosed in bounding rectangles, which represent segmented coins.
5. **Coin Counting**
    - **Counting Segmented Coins:** The total number of valid segmented coin regions is recorded and printed.

### 3.2 Panorama Creation

1. **Keypoint Detection & Descriptor Extraction**
    - **SIFT (Scale-Invariant Feature Transform):** Detects and computes descriptors of local keypoints in each image. This helps to match common features across overlapping images.
2. **Feature Matching**
    - **Brute-Force Matcher:** Matches descriptors from one image to another using a distance metric.
    - **Cross-Check:** Ensures higher-quality matches by confirming that each match is mutually consistent.
3. **Homography Estimation**
    - **RANSAC (Random Sample Consensus):** Used to robustly estimate the homography matrix, which describes how to transform one image so it aligns with another.
4. **Image Warping and Stitching**
    - **Warp Perspective:** Once the homography is found, one image is warped to the coordinate system of the other.
    - **Merging:** The warped image is combined with the reference image to create a panoramic view.

---

## 4. Implementation Details

### 4.1 Repository Structure

```
VR_Assignment1_ShashankVyas_MT2024141/
├── coin_detection.py
├── panorama_creation.py
├── README.md
├── images/
│   ├── coinsun.jpeg
│   ├── segmented_coin_1.png
│   ├── segmented_coin_2.png
│   ├── segmented_coin_3.png
│   ├── segmented_coin_4.png
│   ├── Coin_Task_output.png
│   ├── panroma1.jpeg
│   ├── panroma2.jpeg
│   ├── panroma3.jpeg
│   └── stitched_result.jpg
└──MT2024141_ShashankVyas.pdf
└──MT2024141_ShashankVyas.zip
```

- **`coin_detection.py`:** Implements coin detection, segmentation, and counting.
- **`panorama.py`:** Implements panorama creation using SIFT, feature matching, homography estimation, and image stitching.
- **`images/`:** Contains input images (`coinsun.jpeg` for coin detection; `panroma1.jpeg`, `panroma2.jpeg`, `panroma3.jpeg` for panorama) and output images (segmented coins, stitched panorama).

## 4.2 Software & Libraries

- **Python Version:** 3.x
- **Libraries:**
    - **OpenCV:** For image processing (Install with `opencv-python` and `opencv-contrib-python` for SIFT).
    - **NumPy:** For array manipulations.
    - **Matplotlib:** For visualization.

**Installation Example:**

pip install opencv-python opencv-contrib-python numpy matplotlib

## 4.3 Running the Code

**Coin Detection & Segmentation**

python coin_detection.py

1.
    - **Input:** `coinsun.jpeg` present in the `images/` folder.

- **Output:** Displays and saves segmented coins, prints total coin count.

**Panorama Creation**

python panorama.py

2.

- **Input:** `panroma1.jpeg`, `panroma2.jpeg`, `panroma3.jpeg` in the `images/` folder.
- **Output:** Displays the final stitched panorama and saves `stitched_result.jpg`.

---

# 5. Results

## 5.1 Coin Detection & Segmentation

- **Detected Coins:** The script successfully identifies coin edges and draws contours around each coin.

- **Segmented Coins:**

  - Each coin is extracted based on its bounding box.
  - Example outputs:


**Individual Coin Samples:**


- **Coin Count:**
  if 4 coins are detected, the script prints:
  Total number of coins detected: 4

## 5.2 Panorama Creation

- **Stitched Panorama:**
    - After feature extraction, matching, homography computation, and warping, the final panorama is displayed and saved.
    - Example

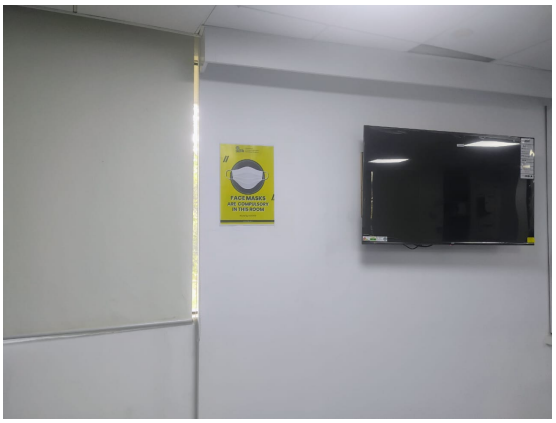
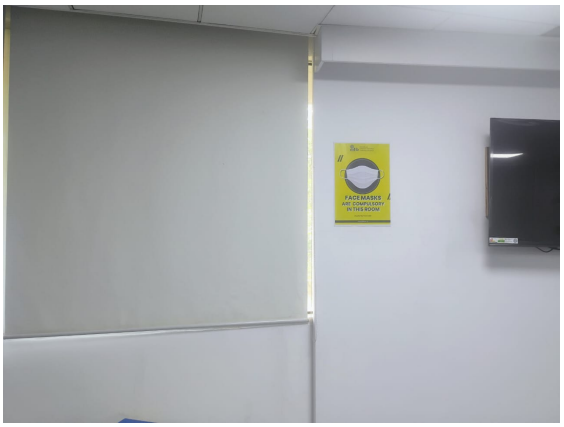
**2 image panorama**

INPUT:

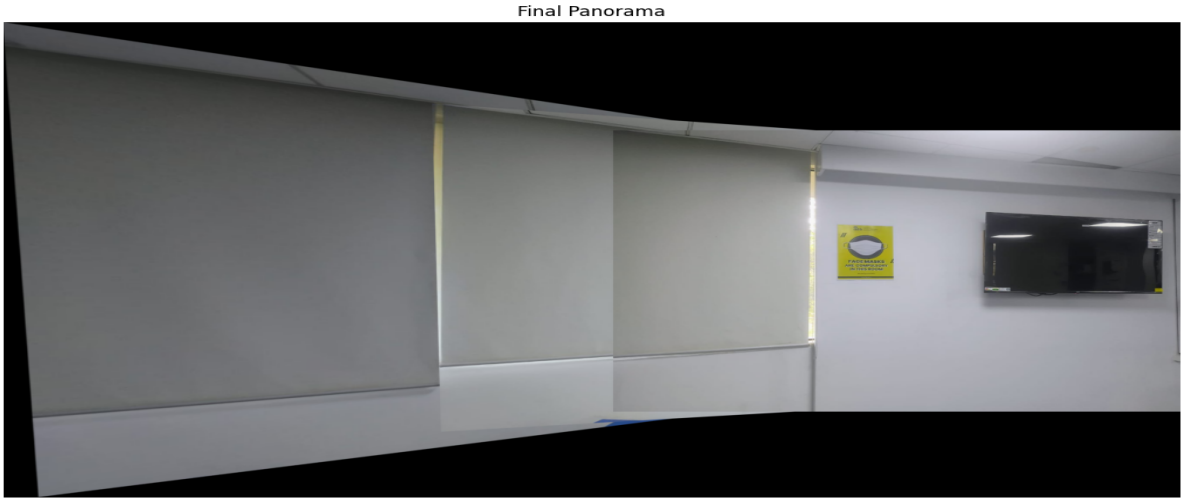

**Panorama Output:**

**3 image panorama**

INPUT:


**Panorama Output:**




---

# 6. Observations

1. **Coin Detection & Segmentation:**
    - **Accuracy:** Proper thresholding in Canny edge detection and filtering contours by area ensures accurate detection.

    - **Challenges:** Very small or large contours may be false positives or missed if area thresholds are not tuned correctly.
    - **Lighting Conditions:** Even lighting helps minimize shadows that could introduce noise or false contours.
2. **Panorama Creation:**

    - **Feature Overlap:** Adequate overlapping regions between images are crucial for successful stitching.
    - **Blending Issues:** Some seams may be visible if there are large exposure differences between images.
    - **Robustness:** The RANSAC algorithm helps handle outliers in feature matching, but good keypoint coverage improves the homography estimation.

---

# 7. Conclusion

- **Coin Detection & Segmentation** provides experience with edge detection, contour analysis, and image segmentation.
- **Panorama Creation** highlights feature-based alignment ,homography matrix and image stitching.

---

GITHUB LINK FOR CODE
https://github.com/Shashank-300/VR_Assignment1_ShashankVyas_MT2024141-