Report

Name:Rohan Rajesh ***Roll No.:***IMT2022575

1. Coin Detection and Segmentation

1.1 Introduction

This task involves detecting and counting coins from an image using various image processing techniques in OpenCV. The approach involves preprocessing, segmentation, contour detection, and filtering to accurately detect the number of coins present.

1.2 Methodology

- **Preprocessing:**
    - Convert the image to grayscale.
    - Apply median blur to reduce noise while preserving edges.
    - Use adaptive Gaussian thresholding to enhance the edges.
- **Edge Detection and Segmentation:**
    - Apply Canny edge detection to extract object boundaries.
    - Perform morphological operations (closing) to fill gaps in detected edges.
    - Use Distance Transform and Connected Components to separate objects.
    - Apply the Watershed algorithm for better segmentation.
- **Contour Detection and Filtering:**
    - Find contours and filter them based on area and circularity to identify coins.
    - Count the filtered contours representing detected coins.
- **Visualization and Output:**
    - Draw the detected coin contours on the original image.
    - Display the number of detected coins on the image.
    - Extract and store segmented coins for further processing.

1.3 Challenges Faced

- **Over-segmentation:** The Watershed algorithm initially produced excessive boundaries.
- **False Detections:** Some non-coin regions were incorrectly identified as coins.
- **Edge Sensitivity:** Finding optimal parameters for Canny edge detection and adaptive thresholding required tuning.

1.4 Final Approach

By refining segmentation with adaptive thresholding and morphological operations, combined with contour filtering based on circularity, false detections were minimized, and accurate coin detection was achieved.

2. Image Stitching and Panorama Creation

2.1 Introduction

This task involves creating a panoramic image by stitching multiple images together. The approach includes feature detection, feature matching, homography estimation, and image blending.

2.2 Methodology

- **Image Loading and Preprocessing:**
    - Load and resize images to optimize computational efficiency.
- **Feature Detection and Matching:**
    - Detect keypoints using the **SIFT (Scale-Invariant Feature Transform) algorithm**.
    - Extract descriptors and match them using the **BFMatcher with a ratio test**.
    - Visualize keypoints and matches for validation.
- **Image Stitching:**
    - Use OpenCV's **Stitcher module** to align and merge images.
    - Apply homography estimation for perspective correction.
- **Post-processing Enhancements:**
    - Adjust contrast and brightness for better visualization.
    - Detect and remove black borders using contour-based cropping.

2.3 Challenges Faced

- **Keypoint Detection Issues:** Some images had fewer keypoints, affecting feature matching accuracy.
- **Perspective Distortions:** Misalignments in stitching were observed in initial trials.
- **Stitcher Errors:** Homography estimation sometimes failed due to poor matches between images.

2.4 Final Approach

- The ratio test for feature matching was fine-tuned to improve accuracy.
- Contrast adjustments were applied before stitching to enhance feature detection.
- The stitched image was cropped to remove unnecessary black borders, improving the final panorama quality.

3. Conclusion

Both implementations successfully achieved their respective objectives:

1. **Coin detection** was improved using adaptive thresholding, morphological operations, and contour filtering.
2. **Panorama stitching** was optimized using improved keypoint matching, homography estimation, and post-processing enhancements.