Coin Segmentation, Counting, and Panorama Stitching

March 2, 2025

1 Introduction

Detecting and counting coins in an image is a common computer vision task with applications in automation, object detection, and image processing. This assignment aims to segment coins from an image and count them. Additionally, we implement an image stitching technique to create a panoramic image from multiple overlapping images.

2 Methodology

2.1 Preprocessing

To enhance the accuracy of coin detection, we performed several preprocessing steps:

- Conversion to **HSV color space** to handle lighting variations.
- Gray color masking to filter out non-coin regions.
- Gaussian blurring to remove noise and enhance edge detection.
- Adaptive thresholding to create a binary image for segmentation.
- Canny edge detection to highlight object boundaries.

2.2 Contour Detection

Contours were detected using `cv2.findContours()`, and we filtered contours based on area and hierarchy to eliminate noise and small artifacts.

2.3 Coin Detection Using Hough Transform

To detect circular shapes, we used the **Hough Circle Transform** with optimized parameters for accuracy. Each detected circle was labeled using `cv2.putText()`.

2.4 Segmentation of Individual Coins

Each coin was extracted using bounding circles and displayed separately.

3 Experiments and Results

The initial approach involved using edge detection with contours, but this resulted in poor segmentation due to noise and false detections. Applying contours on Canny edge detection led to fragmented contours, making it difficult to separate coins properly.

A significant improvement was achieved when we applied contours on the dilated version of the thresholded image. This allowed for better connectivity of the coin boundaries, leading to more accurate segmentation.

Final results:

- Successfully segmented and counted coins in various images.
- Handled lighting variations effectively using HSV conversion.
- Improved accuracy with Hough Circle Transform and adaptive filtering.
- Some false detections still occur due to reflections.

4 Challenges and Limitations

- Bright reflections on shiny coins caused false detections.
- Overlapping coins made individual segmentation difficult.
- Parameter tuning for Hough Transform was required for different images.

5 Panorama Stitching

In addition to coin segmentation, we implemented an image stitching pipeline to create a panoramic image from multiple overlapping images.

5.1 Keypoint Detection and Matching

We used the **SIFT** (Scale-Invariant Feature Transform) algorithm to extract keypoints and descriptors. To match these keypoints between images, we employed the **Brute-Force Matcher** with L2 distance.

5.2 Homography Estimation and Image Warping

Using the matched keypoints, we computed the **homography matrix** using the RANSAC algorithm. This matrix was then used to warp one image onto another, aligning them in a common plane.

5.3 Stitching Process

We loaded three overlapping images and applied the following steps:

- Detect keypoints in each image using SIFT.
- Match keypoints using the Brute-Force Matcher.
- Compute the homography matrix using matched points.
- Warp and blend images to create a seamless panorama.

5.4 Results and Observations

- Successfully stitched three images into a single panoramic output.
- SIFT keypoint detection provided robust feature extraction.
- Homography estimation effectively aligned images with minor distortions.
- Some artifacts were observed at the image boundaries, which could be refined with blending techniques.

6 Conclusion and Future Work

This project successfully implemented a pipeline for detecting and counting coins using OpenCV. Additionally, we performed panorama stitching using keypoint detection and homography estimation. Future improvements could involve:

- Refining morphological operations to handle reflections.

- Enhancing panorama blending to reduce visible seams.

7 References

- OpenCV Documentation: `https://docs.opencv.org/`
- Image Processing Techniques: `https://scikit-image.org/`