VR Assignment -1

Daksh Rajesh
IMT2022019

Edge Detection and Coin Counting

Making the code:

I encountered several issues with edge detection and counting as the code worked when the images were taken from a certain angle but not otherwise.I also faced problem with the detection of 10 rupee coin since it has two colors. Used filters to sharpen image edges and
black and white filters to enhance the results.
Finally used gray scaling and gaussian blur for the best results then canny finally for edge detection.

Objective:

The first part of the assignment focuses on detecting the edges of coins in an image, segmenting them to observe them separately and finally counting the total number of coins in the image.

Methodology:

Image Preprocessing:

- The image is loaded using OpenCV (cv2.imread()) and resized for better visibility.
- It is then converted to grayscale to simplify edge detection.

Noise Reduction:

- A Gaussian blur is applied to smooth the image and reduce noise, improving edge detection.

Edge Detection:

- The **Canny Edge Detector** is used to identify the edges in the image.

Morphological Operations:

- **Dilation** is applied to enhance the detected edges.
- **Closing operation** is performed to fill small gaps and make the shapes more defined.

Contour Detection and Counting:

- The function cv2.findContours is used to detect closed shapes (coins).
- The number of detected contours is counted, representing the total number of coins in the image.

Panorama Stitching Using Feature Matching

Making the code:

I have worked on the exact same problem in my PE done under prof. Behera. But there we used opencv stitcher class which directly stitches the images using the stitcher object created by the function cv2.Stitcher_create(). Here I used sift and orb feature matching approach as well which gave slightly worse results .

Objective:

The second part of the assignment involves stitching overlapping images into a panoramic image using feature matching techniques taught in class (SIFT and ORB).

Methodology:

Image Loading:

- The script loads images from the panorama photos folder using OpenCV (cv2.imread).
- The images are stored in a list for further processing.

Feature Detection:

- Keypoints and descriptors are extracted using SIFT or ORB, selected dynamically for flexibility.

Feature Matching:

- Brute-Force Matcher (BFMatcher) finds feature correspondences between consecutive images.
- Lowe's ratio test filters out weak matches, ensuring only strong correspondences are used.

Homography Computation:

- cv2.findHomography() estimates the transformation between overlapping images.
- RANSAC eliminates incorrect matches to improve alignment accuracy.
- A cumulative homography matrix aligns all images into a single panoramic frame iteratively.

Image Warping & Stitching:

- Each image is warped using its homography transformation with cv2.warpPerspective().
- A blank panorama canvas is created, and the warped images are blended into it.
- A binary mask ensures only valid pixel areas are transferred onto the final panorama.

Black Border Removal Techniques:

- Cropping Method: Detects the largest non-black region using contour detection (cv2.findContours) and removes excess black space.
- Inpainting Method: Reconstructs missing areas by estimating surrounding pixel values using OpenCV's inpainting (cv2.inpaint()).
- Blending Method: Smooths transitions between stitched images to reduce harsh black edges (cv2.addWeighted()).

Output Generation:

- The final processed panorama is saved in part2_stitching_method2_output.
- Three versions of the stitched image are generated:
    - Cropped version: panorama_result_crop.jpg
    - Inpainted version: panorama_result_inpaint.jpg
    - Blended version: panorama_result_blend.jpg