Assignment 1

Pradyumna G – IMT2022555

(Github link: https://github.com/PradyumnaG/VR_Assignment1)

Overview:

1. Coin Detection:

- Detect coins using edge detection.
- Identify number of coins.
- Seperate each individual coin.
- Segment coins.

2. Image stitching for panorama creation

- Detect Key Point descriptors.
- Identify common Key Point descriptors.
- Stich the images to produce a final panorama.

Coin Detection:

1. Image Preprocessing:

- Resizing ensures consistent processing for images of varying sizes.
- Grayscale conversion simplifies the image to a single channel.
- Gaussian blur reduces noise, making the thresholding step more effective.
- Adaptive thresholding is used.

2. Coin Detection:

- Contours are extracted using cv2.findContours.
- Circularity is calculated to determine how close a contour is to a perfect circle.
- Contours with circularity between 0.7 and 1.2 and an area above a minimum threshold are considered valid coins.

3. Segmentation and Masking:

- **Outline mode**: Draws the contours of the detected coins on the original image.
- **Mask mode**: Creates a binary mask where the detected coins are filled with a specific color (orange in our case).
- The coins are extracted from the previous step using contours.

4. Individual Coin Extraction:

- A circular mask is created for each coin using cv2.minEnclosingCircle to determine the center and radius.
- The mask is applied to the original image to isolate the coin.
- The coin is cropped using the bounding box of the circle and saved as a separate file.

Requirements:

pip install numpy opencv-python

How to run:

Place all your input coin images (name can be anything) in Assignment1/Part1/Input. Run the file using the command: python3 CoinDetectionOrange.py

Panorama Creation:

1.Feature Detection:

- cv2.SIFT_create() initializes the SIFT detector.
- sift.detectAndCompute() detects keypoints and computes their descriptors (feature vectors).
- SIFT keypoints are calculated for both the images.

2. Feature Matching:

- Matches the descriptors of keypoints from the two images using brute force.
- cv2.BFMatcher() initializes the matcher with the L2 norm (Euclidean distance) for comparing descriptors.
- matcher.match() finds the best matches between the descriptors of the two images.
- Matches are sorted by distance (quality of match), and only the top matches are used for further processing.

3. Image Stitching:

- cv2.findHomography() computes the homography matrix using the RANSAC algorithm to handle outliers.
- cv2.warpPerspective() warps the first image to align it with the second image using the homography matrix.
- The second image is then overlaid on the warped image to create the final panorama.

Requirements:

pip install opencv-python numpy

How to run:

Place the input images in Assignment1/Part2/input and replace the input paths in the code.

Observations and Results:

Coin Detection:

- Successfully detects coin edges using contour edge detection.

- Successfully detects the number of coins and extracts each individual coin.

- Effectively segments all the coins and generates a mask of all the coins.

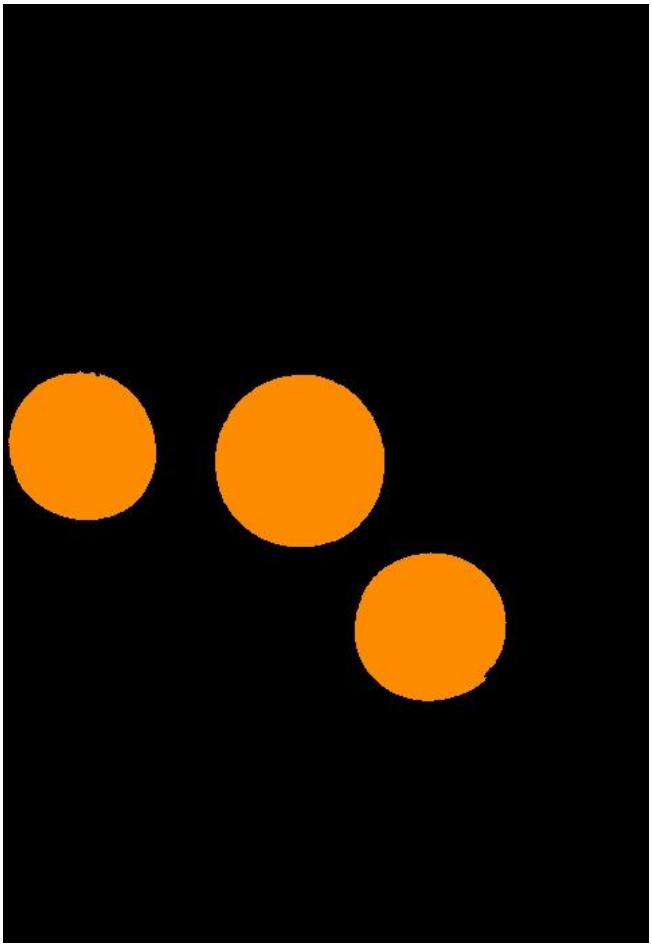

Image stitching:

- SIFT keypoints are accurately detected in both the images.

- Keypoints/Features are accurately matched between both the images.

- The final panorama image is successfully generated using RANSAC algorithm and homography.