Visual Recognition AIM-825

Assignment 1 Report

Manda Kausthubh (IMT2022027)[a]

[a]*IIITB, Electronics City Phase 1, Hosur Road, Bengaluru - 560100.*

March, 2025

Abstract

Assignment-1 deals with the topics of Feature Extraction and Non-ML methods in Computer Vision. This includes 2 tasks: 1. **Coin detection**: Given an image of coins can you detect the coins, count the total number of coins and segment them in a giving image. 2. **Image Stitching**: Stitch images from two images which are adjacent to each other and combine them to make single image that is continuous.

In this report we discuss various methods that I have tried to reach the required solutions and compare these methods.

Figure 1: Cut outs for a 2 Rupee coins from an image

1. Introduction

Computer vision is a rapidly evolving field that enables machines to interpret and analyze visual data. While modern advancements heavily rely on machine learning (ML) techniques, several classical non-ML approaches remain effective for many fundamental computer vision tasks. These traditional methods leverage mathematical models, geometric transformations, and image processing techniques to extract meaningful information from images without requiring extensive training on large datasets.

1.1. Problem Description

In this report, we explore non-ML methods for solving specific computer vision tasks, such as coin counting and image stitching. Coin counting, widely used in automated vending machines and banking applications, typically involves techniques like edge detection, contour analysis, and Hough Circle Transform to identify and count coins in an image. Similarly, image stitching, a crucial process in panoramic photography and medical imaging, relies on feature detection algorithms such as SIFT (Scale-Invariant Feature Transform) to align and merge multiple images into a seamless composite.

By analyzing these approaches, we aim to highlight the robustness and efficiency of classical computer vision techniques in scenarios where ML-based solutions may be computationally expensive, data-dependent, or impractical.

2. Theory

2.1. Coin Counting Using Traditional Computer Vision Methods

Coin counting is a fundamental task in computer vision, often used in banking, vending machines, and automated sorting systems. Classical image processing techniques enable coin detection and counting without requiring machine learning models. The key steps include:

2.1.1. Preprocessing

To enhance image quality and remove noise, preprocessing techniques such as grayscale conversion and Gaussian blurring are applied:

- **Grayscale Conversion:** Reduces computational complexity by eliminating color information, making edge detection more efficient.
- **Gaussian Blurring:** Smooths the image to suppress noise and improve edge detection accuracy.

(a) Coin segmentation with Hough Transform

(b) Segmentation using Canny Edge detection

(c) Segmentation using Laplacian Edge detection

(d) Coin segmentation with Hough Transform (American Coins)

(e) Segmentation using Laplacian Edge detection

(f) Segmentation using Canny Edge detection

Figure 2: The various images generated through highlighting using various edge detection and counting methods.

2.1.2. Edge Detection and Contour Analysis

Identifying coin boundaries is crucial for counting. Edge detection algorithms such as the Canny edge detector are commonly used:

- **Canny Edge Detection:** A multi-stage algorithm that detects edges by finding intensity gradients.
- **Contour Detection:** Extracts closed boundary curves from the detected edges, which can be used to identify individual coins.

2.1.3. Circle Detection Using Hough Transform

Coins are circular objects, making the Hough Circle Transform (HCT) an effective method for detection:

$$x = a + r\cos\theta, \quad y = b + r\sin\theta \tag{1}$$

where (a, b) represents the circle center, r is the radius, and θ varies from 0 to 2π. The HCT detects circles by mapping edge points in image space to a parameter space.

2.2. Image Stitching Using Feature-Based Methods

Image stitching is the process of combining multiple overlapping images to create a seamless panorama. The approach involves key steps such as feature detection, matching, homography estimation, and image warping.

2.2.1. Feature Detection Using ORB

The **Oriented FAST and Rotated BRIEF (ORB)** algorithm is used for detecting and describing keypoints in both images. ORB is computationally efficient and invariant to scaling and rotation. The keypoints and descriptors are extracted as follows:

```
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
```

2.2.2. Feature Matching Using Brute-Force Matcher

Once features are detected, they are matched across images:

```
bf = cv2.BFMatcher(cv2.NORM_HAMMING,
                   crossCheck=True)
matches = bf.match(des1, des2)
matches = sorted(matches,
                 key=lambda x: x.distance)
```

The **Brute-Force Matcher** directly compares descriptors and selects the best matches based on the minimum distance.

2.2.3. Homography Estimation

A **homography matrix** transforms points from one image to another. Using the matched feature points, the homography is computed:

```
src_pts = np.float32([kp1[m.queryIdx].pt
                      for m in matches]).reshape(-1, 1, 2)
dst_pts = np.float32([kp2[m.trainIdx].pt
                      for m in matches]).reshape(-1, 1, 2)
H, mask = cv2.findHomography(src_pts,
                             dst_pts, cv2.RANSAC, 5.0)
```

The **RANSAC (Random Sample Consensus)** algorithm eliminates outlier matches and ensures a robust transformation.

Mathematically, the homography transformation is represented as:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{2}$$

where H is a 3×3 matrix mapping points from one image to another.

2.2.4. Image Warping and Blending

Using the computed homography, the first image is warped to align with the second:

```
h, w, _ = img2.shape
img1_warped =
    cv2.warpPerspective(img1,
        H, (w * 2, h))
```

Finally, the images are merged to create a seamless panorama:

```
img1_warped[0:h, 0:w] = img2
```

This classical approach to image stitching is efficient, robust, and widely used in applications like panoramic photography and medical imaging.

3. Method and Equipment

3.1. Coin Counting Using Traditional Computer Vision

The process of coin counting involves detecting circular objects in an image using classical image processing techniques. The following steps outline the approach:

3.1.1. Step 1: Preprocessing

To improve the quality of the image and remove noise, we apply preprocessing techniques:

```
image = cv2.imread("coins.jpg")

gray = cv2.cvtColor(image,
            cv2.COLOR_BGR2GRAY)


blurred = cv2.GaussianBlur(gray,
                           (11,11), 0)
```

- The image is converted to **grayscale** to reduce complexity. - **Gaussian blurring** is used to remove noise and smooth the image.

3.1.2. Step 2: Edge Detection and Contour Detection

```
edges = cv2.Canny(blurred, 30, 150)

contours, _ = cv2.findContours(edges,
        cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
```

- **Canny edge detection** is applied to highlight coin boundaries. - **Contour detection** extracts closed curves representing each coin.

3.1.3. Step 3: Circle Detection Using Hough Transform

```
circles = cv2.HoughCircles(gray,
                           cv2.HOUGH_GRADIENT,
                           1, 20,
                           param1=50,
                           param2=30,
                           minRadius=10,
                           maxRadius=100)

if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        cv2.circle(image, (i[0], i[1]),
                   i[2], (0, 255, 0), 3)
```

- The **Hough Circle Transform** detects circular shapes corresponding to coins. - The detected coins are drawn on the image.

3.2. Image Stitching Using Feature-Based Methods

Image stitching involves aligning and blending multiple images into a seamless composite. The following steps describe the methodology:

3.2.1. Step 1: Feature Detection

```
img1 = cv2.imread("1.jpg")
img2 = cv2.imread("2.jpg")

orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
```

- The **ORB (Oriented FAST and Rotated BRIEF)** algorithm detects and describes key features in both images.

3.2.2. Step 2: Feature Matching

```
bf = cv2.BFMatcher(cv2.NORM_HAMMING,
                   crossCheck=True)
matches = bf.match(des1, des2)

matches = sorted(matches,
        key=lambda x: x.distance)
```

3.2.3. Step 3: Homography Estimation

```
src_pts = np.float32([kp1[m.queryIdx].pt
            for m in matches]
            ).reshape(-1, 1, 2)
dst_pts = np.float32([kp2[m.trainIdx].pt
            for m in matches]
            ).reshape(-1, 1, 2)

H, mask = cv2.findHomography(src_pts,
                   dst_pts,
                   cv2.RANSAC, 5.0)
```

- The **homography matrix** is estimated using the RANSAC algorithm to align images.

3.2.4. Step 4: Image Warping and Blending

```
h, w, _ = img2.shape
img1_warped = cv2.warpPerspective(img1,
                                  H,
                                  (w * 2, h))

img1_warped[0:h, 0:w] = img2
```

- The first image is **warped** to align with the second image. - The images are **blended** into a seamless composite.

4. Results

4.1. Coin Counting

The coin counting approach successfully detected and counted circular objects in the test images. The results were evaluated based on detection accuracy and robustness against noise.

4.1.1. Detected Coins and Accuracy

The Hough Circle Transform effectively identified circular objects, with minimal false detections. An example result is shown in Figure 3, where green circles represent detected coins. To evaluate the accuracy, the number of detected coins was compared against the actual count. Table 1 summarizes the results.

Figure 3: Detected coins using Hough Circle Transform

Image	Actual Coins	Detected Coins
Sample 1	2	2 (Canny)
Sample 2	4	4 (Hough, Canny)
Sample 3	8	8 (Hough)

Table 1: Comparison of actual and (best) detected coin counts

The approach demonstrated by Hough Transformer and Canny seem to be the most promising, with minor misdetections occurring due to overlapping coins or poor lighting conditions.

4.1.2. Effect of Noise on Detection

To analyze robustness, Gaussian noise was added to images, and detection performance was evaluated. The Hough Transform remained effective under moderate noise levels but degraded under severe noise conditions. Preprocessing steps such as adaptive thresholding improved performance in such cases. We can also note that any attempt which involves repeatedly applying blurring (Gaussian) followed by same amount of image sharpeing although effective in damping the noise of the markings inside the coin, however it also results in addition of more features resulted blurring and creation of pockets on around the true boundries of the coin edges, and hence is not suitable.

4.2. Image Stitching

The feature-based image stitching method successfully aligned and blended overlapping images to form a seamless panorama. The effectiveness of stitching was evaluated based on feature matching accuracy and visual consistency.

4.2.1. Feature Matching Results

The ORB feature detector efficiently identified keypoints in both images. Figure 4 illustrates the matched features between two input images.

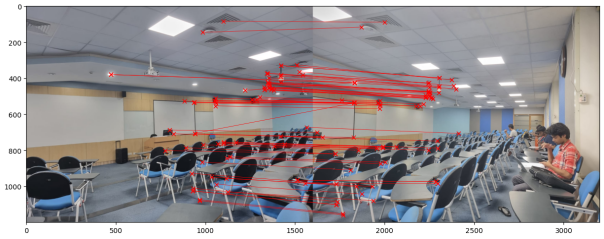

Figure 4: Feature matching using ORB and Brute-Force Matcher

The number of correctly matched keypoints was analyzed to determine performance.

4.2.2. Stitched Image Quality

The homography transformation aligned the images accurately, and the final panorama showed minimal distortion. Figure 5 presents an example of the stitched output.

Figure 5: Final stitched image

4.2.3. Challenges and Limitations

- **Blending Artifacts:** Minor intensity differences at image boundaries resulted in visible seams.
- **Perspective Distortion:** Large viewpoint differences between images affected alignment.
- **Low-Texture Regions:** Feature detection struggled in images with uniform backgrounds.

4.3. Summary

The results demonstrate that traditional computer vision techniques provide effective solutions for coin counting and image stitching. Coin detection achieved high accuracy, while feature-based image stitching produced seamless panoramas with minimal artifacts. Some limitations were observed in cases of severe noise, non-uniform lighting, and extreme perspective variations.

5. Conclusion

The conclusion should summarise your main results and main points from the discussion. A rule of thumb is to not present any new information (information not found in the results or discussion).

Acknowledgements

Some of the code especially regarding SIFT and ORB for Image stitching was taken and inspired from: Link.

I would also like to mention that the coin images were borrowed from Jinesh Pagaria (IMT2022044).