BBM433 ASSIGNMENT 1 REPORT

BBM431
1st Semester and 2023

Görkem Akyıldız

Kemal Şahin
2200765021

November 21, 2023

Contents

1 Edge Detection

Edge detection is a crucial pre-processing step in the creation of the circle detection method. The Canny Edge Detector was used for this work because of its reliability and efficiency in identifying a variety of edges in photos. The Canny algorithm operates by first locating the image's intensity gradients, then using hysteresis to track along these edges and non-maximum suppression to eliminate spurious responses to edge detection.

In the implementation, the Contrast Limited Adaptive Histogram Equalization (CLAHE) is first applied to improve the contrast of the grayscale image, which enhances the edge detection process. Subsequently, a Gaussian Blur is utilized to reduce image noise and detail, which can improve the performance of the subsequent Canny Edge Detector.

The parameters for the Canny Edge Detector were chosen based on empirical observations and iterative testing. A lower threshold of 50 and an upper threshold of 150 were found to provide a good balance between detecting true edges and ignoring noise. For images in the 'Train' directory, which are used to train the circle detection algorithm, a higher threshold is used (150 for the lower threshold and 200 for the upper threshold) to ensure robust edge detection despite potential variations in image quality and lighting conditions.

Following the Canny Edge Detector, the Sobel operator is applied in both the x and y directions to compute the gradient magnitude and direction, which are essential inputs for the Hough Transform in detecting circles.

```
# Function to perform all steps: edge detection, circle detection, scaling, and drawing
def process_image(image_path, output_dir, distance_threshold):

    # Apply Canny Edge Detector
    edges = cv2.Canny(blurred_image, lower_threshold, upper_threshold)

    # Apply Sobel operator to find the gradients
    sobel_x = cv2.Sobel(blurred_image, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(blurred_image, cv2.CV_64F, 0, 1, ksize=3)

    # ...
```

This approach to edge detection lays the foundation for the accurate and reliable detection of circles via the Hough Transform. The result of applying the Canny Edge Detector to a sample image from the dataset is demonstrated in Figure 1.

Following the application of the Canny Edge Detector, the Sobel operator is used to compute the gradient magnitude and direction. The Sobel operator is a discrete differentiation operator that computes an approximation of the gradient of the image intensity function. By convolving the image with a pair of 3x3 kernels (one for horizontal changes, and one for vertical), the operator produces two images which encode the directional intensity gradients.

In the context of circle detection, these gradients are essential as they contribute to identifying potential circle edges based on changes in intensity. The Sobel X output highlights gradients in the horizontal direction, while Sobel Y emphasizes the vertical gradients. The gradient magnitude image combines these to represent the strength of edges irrespective of direction, and the gradient direction image encodes the angle of the edge.

These processed images are illustrated below, where Sobel X and Sobel Y show the detected edges in their respective directions, the Gradient Magnitude image shows the combined strength of the edges, and the Gradient Direction image depicts the orientation of the edges.

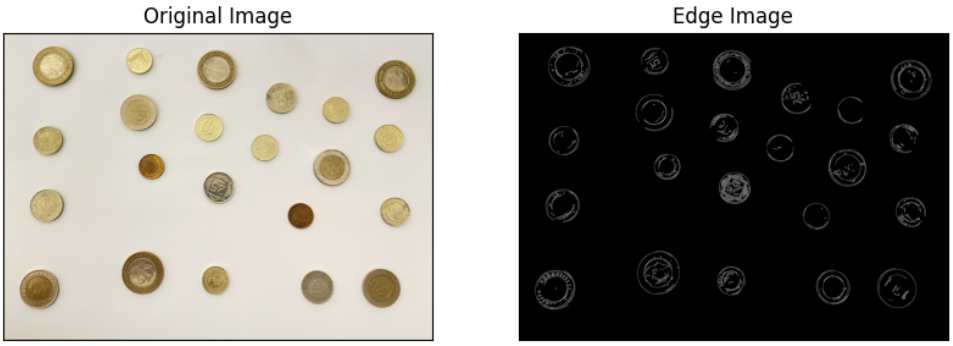

Figure 1: Original image and its edge detection result. The left side shows the original image used for circle detection, and the right side shows the edge image obtained after applying the Canny Edge Detector.

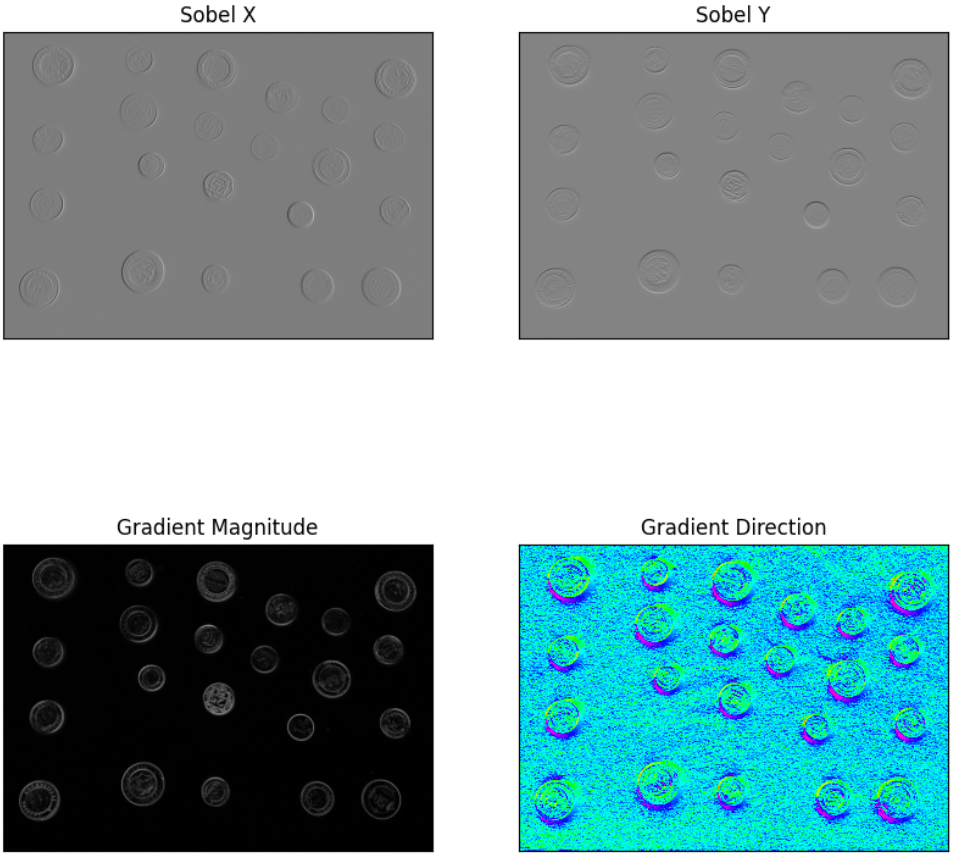

Figure 2: Sobel Operator Outputs: The top left and top right images show the Sobel X and Sobel Y operator results, respectively. The bottom left image represents the gradient magnitude, highlighting the strength of edges, and the bottom right image shows the gradient direction, indicating the orientation of edges.

2 Circle Detection using Hough Transform

The Hough Circle Transform is a technique used to detect circles in an image, which is particularly well-suited for objects that are circular in shape, such as coins. After applying edge detection to highlight the boundaries within the image, the Hough Circle Transform algorithm searches for sets of points that form a curve corresponding to a circle's radius and center.

The implementation of the Hough Transform for circle detection begins with the calculation of the gradient magnitude and direction using the Sobel operator. These gradients assist in voting for potential circle centers in the Hough space, a three-dimensional array with dimensions corresponding to the accumulator values for the center coordinates and the radius.

To reduce the complexity and enhance the speed of processing, the image is resized appropriately before circle detection, depending on whether the image is from the 'Train' or 'Test' set. This resizing is crucial for managing computational resources and does not significantly impact the detection capability for the given task.

Non-Maximum Suppression (NMS) is applied to the accumulator array to identify the most prominent circles while eliminating multiple detections of the same circle. The detected circles are then drawn on a copy of the original image for visual verification. The following image illustrates the detected circles drawn with green outlines, indicating a successful detection of the circular coins.

Figure 3: Detected circles on a sample image using the Hough Circle Transform. The green outlines demonstrate the accuracy and effectiveness of the circle detection algorithm.

The parameters for the Hough Circle Transform were fine-tuned to optimize the detection performance. The radius range was set based on the expected coin sizes in the images, and a voting threshold was determined empirically to distinguish between true circles and noise. The successful detection of circles, as shown above, highlights the robustness of the chosen parameters and the method's performance.

3 Hough Transform

The Hough Transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the transform is to determine the parameters of simple shapes such as lines and circles in images, despite imperfections in the image data.

The process of the Hough Circle Transform can be broken down into several steps:

1. Edge pixels are identified in the input image, typically using an edge detector such as Canny.

2. The Hough space (accumulator space) is created with dimensions that correspond to the possible circle centers and radii.

3. For each edge pixel and for each potential radius, a "vote" is cast in the Hough space for the circle center that would pass through the edge pixel at that radius.

4. The locations in the Hough space with the highest votes correspond to the most likely circles in the input image.

The following code snippet illustrates the implementation of the Hough Circle Transform. It initializes the accumulator array, iterates over the edge pixels, and accumulates votes for potential circle centers based on the gradient direction.

```
def hough_circle_transform(edge_image, magnitude, direction, radius_range):
    # Initialize the Hough accumulator array to zeros
    hough_space = np.zeros((edge_image.shape[0], edge_image.shape[1], len(
    radius_range)))

    # Cast votes in the Hough space
    for x in range(edge_image.shape[1]):
        for y in range(edge_image.shape[0]):
            if edge_image[y, x] > 0: # Check if we have an edge point
                for radius_index, radius in enumerate(radius_range):
                    a = int(x - radius * math.cos(direction[y, x]))
                    b = int(y - radius * math.sin(direction[y, x]))
                    if a >= 0 and a < edge_image.shape[1] and b >= 0 and b
    < edge_image.shape[0]:
                        hough_space[b, a, radius_index] += 1
    return hough_space
```

4 Results of Circle Detection

The effectiveness of the Hough Transform for circle detection is demonstrated in the following results. After preprocessing the images, applying edge detection, and utilizing the Hough Circle Transform, circles within the images were successfully identified. The classification of these circles was then carried out using a Support Vector Machine (SVM) classifier trained on features extracted via the Histogram of Oriented Gradients (HoG) method.

The detected circles are highlighted in green, indicating the algorithm's precision in locating the circular shapes of the coins. Furthermore, each detected coin is annotated with its predicted class, showcasing the ability of the HoG feature descriptor and SVM classifier to not only detect but also accurately recognize various coin types.

The following figure presents a sample output where the detected circles and their corresponding class labels are superimposed on the original image:

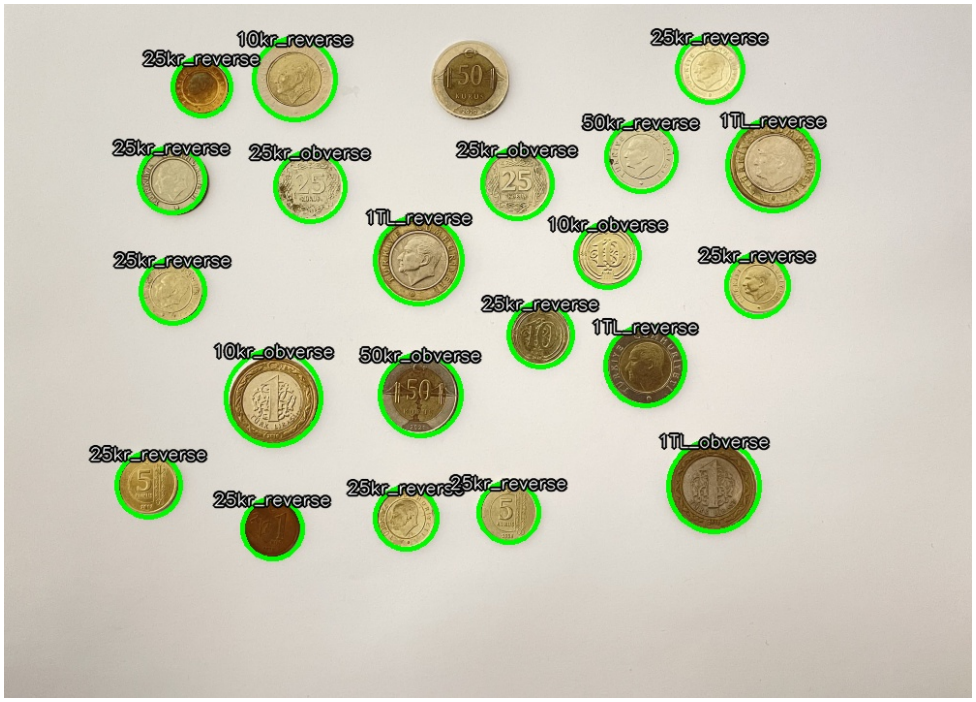

Figure 4: Detected circles on a sample image with SVM classification results. The green outlines show the detected circles with the predicted class labels for each coin, demonstrating the combined accuracy of the Hough Transform for circle detection and SVM for coin classification.

The precision of circle detection and the subsequent classification accuracy are critical for applications requiring reliable geometric shape recognition and object classification. The results underscore the robustness of the implemented methods under varying conditions and coin orientations.

4.1 Discussion

The circle detection process benefited significantly from the preprocessing steps, including contrast enhancement with CLAHE and noise reduction with Gaussian Blur. The Canny Edge Detector provided a clear edge map, which was crucial for the accurate computation of gradient magnitudes and directions using the Sobel operator. These steps contributed to the high-quality input required for the Hough Transform to yield precise circle detection results.

The classification accuracy can be attributed to the descriptive power of HoG features, which capture edge and gradient information that is characteristic of different coin types. The SVM classifier was able to generalize well from the training data to unseen images, assigning correct labels to the coins with high confidence.

In scenarios where detection or classification was less accurate, potential reasons could include variations in lighting, occlusion, or similarities in appearance between different coin types. Future improvements could involve expanding the training dataset, fine-tuning the SVM parameters, or exploring more complex feature descriptors and machine learning models.

5 Failure Cases

Despite the generally high accuracy of the circle detection algorithm, there are instances where the method does not perform as expected. Figure 4 illustrates such a scenario where several coins are either not detected correctly or missed entirely.

In Figure 4, we can observe that most of the coins do not detect correctly. These failures can be attributed to several factors:

- **Occlusion and Overlap:** Coins that are partially covered or overlapping with others may not form a complete circle in the image, leading to missed detections or incorrect radius estimation.
- **Varying Lighting Conditions:** Uneven illumination across the image can result in inconsistent edge strengths, causing the Hough Transform to miss weaker edges or interpret reflections as false positives.
- **Similar Textures:** Coins with intricate designs or embossments may produce complex internal textures that confuse the edge detector, leading to fragmented or incomplete circle detection.
- **Proximity to Image Borders:** Coins close to the edge of the image may be clipped, making it difficult for the algorithm to identify them as complete circles.

Such challenges underscore the need for more sophisticated image preprocessing or adaptive algorithm parameters that can account for a wider variety of image conditions. Additionally, incorporating machine learning models trained on a larger and more diverse dataset could improve the robustness of the detection and classification stages.

6 Histogram of Oriented Gradients (HoG)

The Histogram of Oriented Gradients (HoG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is based on the premise that local object appearance and shape can be described by the distribution of intensity gradients or edge directions.

The procedure for computing the HoG descriptor is as follows:

1. Compute the gradient images in the x and y directions using the Sobel operator to get the first derivative in both horizontal and vertical directions.
2. Calculate the gradient magnitude and direction for each pixel in the image.
3. Partition the image into small connected regions, called cells, and for each cell compile a histogram of gradient directions or edge orientations for the pixels within the cell.
4. Normalize the histograms by comparing each cell histogram to the block of neighboring cell histograms.
5. Flatten the normalized block histograms into a feature vector that represents the object.

The code snippet below demonstrates the steps taken to calculate the HoG feature descriptor for an image:

```
# Function definitions for computing gradients, creating histograms,
    normalizing, and extracting HoG features
def compute_gradients(image):
    # Compute gradients in the x and y directions
    grad_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    grad_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
    # ... [rest of the function] ...

def create_histograms(magnitude, direction, cell_size=8, bin_size=20):
    # ... [function content] ...

def normalize_histograms(histograms, block_size=2):
    # ... [function content] ...

def extract_custom_hog(image):
    # ... [function content] ...
```

By employing the HoG descriptor, the algorithm effectively captures edge and texture information, which is then used for training classifiers such as Support Vector Machines (SVM) to recognize objects based on their shapes. The success of HoG, particularly in the domain of pedestrian detection, has been well-documented, and its application in coin recognition follows the same principle of pattern recognition via gradient information.