CV Assignment 2

Rawan Farouk 16001938

November 25, 2024

1 Methodology of Each Algorithm

1.1 Moravec Corner Detection

Moravec's method calculates the variations in intensity within a small window along multiple directions-horizontal, vertical, and diagonal. A point is considered to be a corner if the minimum variation in intensity across these shifts is greater than some pre-specified threshold.

1.2 Harris Corner Detection

Harris' method uses image gradients to compute the structure tensor for determining cornerness by evaluating the eigenvalues of the tensor matrix. The corners are identified as points that have a high intensity variation in all directions, hence yielding more accuracy and robustness than Moravec's method.

2 Python Implementation with Details

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the chess image
image = cv2.imread('chess.png', cv2.IMREAD_GRAYSCALE)

# Function: Moravec Corner Detection

1.Define a small window size (e.g., 3x3) to analyze intensity variations.
2.For each pixel in the image (excluding boundary regions):
   - Extract a window centered at the current pixel.
   - Compute intensity differences by shifting the window in four directions: right, left,
      diagonal down-right, and diagonal down-left.
   - Measure the sum of squared differences for each shift.
   - Identify the pixel as a corner if the minimum intensity difference exceeds a
      threshold.
3.Append the detected corner coordinates to the corners list.
```

```
def moravec_corner_detection(image, threshold=100):
    corners = []
    height, width = image.shape
    window_size = 3
    offset = window_size // 2

    for y in range(offset, height - offset):
        for x in range(offset, width - offset):
            region = image[y-offset:y+offset+1, x-offset:x+offset+1]
            intensities = [
                np.sum((region - np.roll(region, shift, axis=axis))**2)
                for shift, axis in [(1, 0), (-1, 0), (1, 1), (-1, 1)]
            ]
            if min(intensities) > threshold:
                corners.append((x, y))
    return corners
```

```
1.Compute image gradients along the x and y axes using Sobel filters.
2.Construct the structure tensor components (Ixx, Iyy, Ixy) by squaring and multiplying
  the gradients.
3.Smooth the components using a Gaussian filter to reduce noise.
4.Calculate the Harris response at each pixel.
5.Identify corners as pixels where the Harris response exceeds a threshold.
6.Store detected corners in the corners list.
```

```
# Function: Harris Corner Detection
def harris_corner_detection(image, k=0.04, threshold=1e6):
    # Compute gradients
    Ix = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    Iy = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)

    # Compute structure tensor components
    Ixx = Ix**2
    Iyy = Iy**2
    Ixy = Ix * Iy

    # Apply Gaussian filter to smooth
    Ixx = cv2.GaussianBlur(Ixx, (3, 3), 1)
    Iyy = cv2.GaussianBlur(Iyy, (3, 3), 1)
    Ixy = cv2.GaussianBlur(Ixy, (3, 3), 1)

    # Harris response calculation
    harris_response = (Ixx * Iyy - Ixy**2) - k * (Ixx + Iyy)**2

    # Threshold and extract corners
    corners = np.argwhere(harris_response > threshold)
    return corners[:, [1, 0]] # Reverse (y, x) to (x, y)

# Detect corners using both methods
```

```
moravec_corners = moravec_corner_detection(image, threshold=100)
harris_corners = harris_corner_detection(image, threshold=1e6)

# Visualization: Moravec Corners
plt.figure(figsize=(10, 10))
plt.imshow(image, cmap='gray')
plt.scatter(*zip(*moravec_corners), color='red', s=10, label='Moravec Corners')
plt.legend()
plt.title('Moravec Corner Detection')
plt.show()

# Visualization: Harris Corners
plt.figure(figsize=(10, 10))
plt.imshow(image, cmap='gray')
plt.scatter(harris_corners[:, 0], harris_corners[:, 1], color='green', s=10, label='Harris
    Corners')
plt.legend()
plt.title('Harris Corner Detection')
plt.show()

# Visualization: Both Corners Together
plt.figure(figsize=(10, 10))
plt.imshow(image, cmap='gray')
plt.scatter(*zip(*moravec_corners), color='red', s=10, label='Moravec Corners')
plt.scatter(harris_corners[:, 0], harris_corners[:, 1], color='green', s=10, label='Harris
    Corners')
plt.legend()
plt.title('Corner Detection (Moravec: Red, Harris: Green)')
plt.show()
```

3 Results

3.1 Moravec Detection

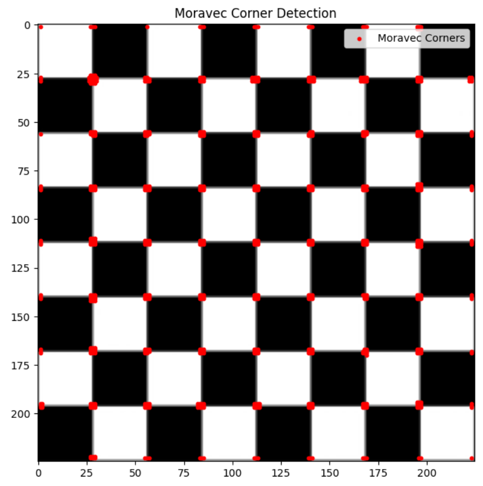

Figure 1: Moravec Detection

3.2 Harris Detection

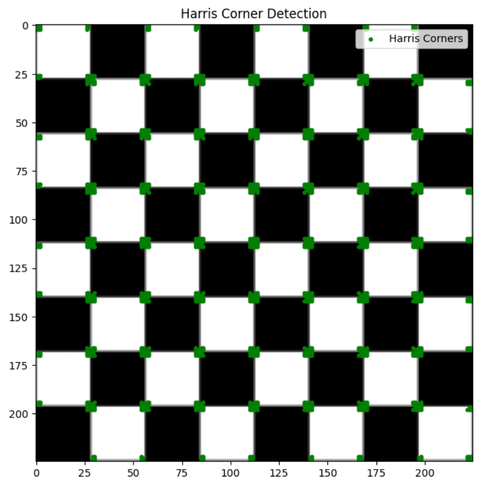

Figure 2: Harris Detection

3.3 Harris Vs Moravec Detection

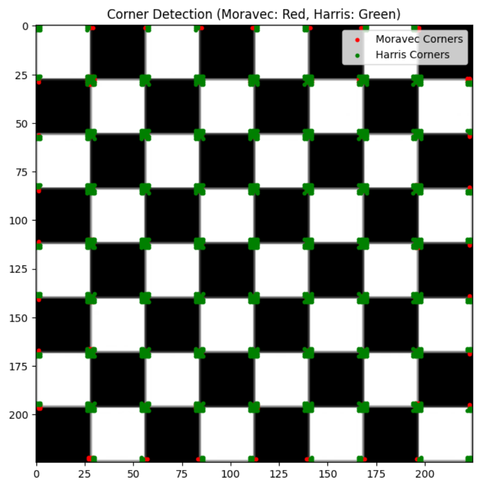

Figure 3: Harris vs Moravec

4 Discussion

4.1 Number of Detected Corners

- **Moravec** detects fewer corners since it is sensitive to variations in intensity along any particular direction.
- **Harris** has eigenvalue-based calculations that capture finer details, hence facilitating the detection of more corners.

4.2 Corner Distribution

- **Moravec corners** are rare and are found only for very distinctive corner-like features.
- **Harris** gives more uniform distribution in regions of high variation of gradient and even detects subtle corner-like features.

4.3 Robustness to Noise

- **Moravec** is less robust for noise, as it depends upon the changes in intensity values, which is highly affected by noise.
- **Harris** can be seen as more robust since its smoothing step (Gaussian filter) reduces the effect of noise on the gradient calculations.

4.4 Computational Efficiency

- **Moravec** is computationally simpler and faster due to its reliance on intensity comparisons in only four directions; suited to situations where speed is more important than accuracy.
- **Harris** is computationally more expensive since it involves gradient calculation, structure tensor construction, and eigenvalue computation; more suitable for applications requiring high accuracy and robustness.