Line, Corner, Feature Dscriptor

Krithika , 22011101046

April 7, 2025

Introduction

This document describes the application of various image processing techniques using Python libraries such as OpenCV, Skimage, and Scikit-learn. The tasks include edge and line detection, corner detection, feature detection and matching using SIFT and HoG descriptors.

Python Code

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage.transform import rotate, resize
from sklearn.metrics.pairwise import cosine_similarity

# Load the image
image_path = "1.png"  # Change this to your image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Task 1: Line Detection using Canny and Hough Transform
edges = cv2.Canny(image, 50, 150, apertureSize=3)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=50, minLineLength=20, maxLineGap=5)

line_image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
if lines is not None:
    for line in lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(line_image, (x1, y1), (x2, y2), (0, 255, 0), 2)

# Task 2: Corner Detection using Harris Corner Detection
corners = cv2.cornerHarris(image, 2, 3, 0.04)
corners = cv2.dilate(corners, None)
corner_image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
corner_image[corners > 0.01 * corners.max()] = [0, 0, 255]

# Task 3: Feature Descriptors using SIFT
sift = cv2.SIFT_create()
keypoints, descriptors = sift.detectAndCompute(image, None)
sift_image = cv2.drawKeypoints(image, keypoints, None)

# Task 4: Rotate Image 45 Degrees and Match Features using HoG
rotated_image = rotate(image, 45, resize=True) * 255
rotated_image = rotated_image.astype(np.uint8)
rotated_image = resize(rotated_image, image.shape) * 255
rotated_image = rotated_image.astype(np.uint8)

hog_features_original, _ = hog(image, pixels_per_cell=(8, 8), cells_per_block=(2, 2),
↪  visualize=True)
hog_features_rotated, _ = hog(rotated_image, pixels_per_cell=(8, 8), cells_per_block=(2, 2),
↪  visualize=True)
```

```
min_length = min(len(hog_features_original), len(hog_features_rotated))
hog_features_original = hog_features_original[:min_length].reshape(1, -1)
hog_features_rotated = hog_features_rotated[:min_length].reshape(1, -1)
matching_score = cosine_similarity(hog_features_original, hog_features_rotated)[0][0]

print(f"HoG Feature Matching Score: {matching_score:.4f}")

# Display the results
plt.figure(figsize=(15, 10))
plt.subplot(2, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.axis('off')

plt.subplot(2, 3, 2)
plt.imshow(line_image)
plt.title("Detected Lines")
plt.axis('off')

plt.subplot(2, 3, 3)
plt.imshow(corner_image)
plt.title("Detected Corners")
plt.axis('off')

plt.subplot(2, 3, 4)
plt.imshow(sift_image)
plt.title("SIFT Features")
plt.axis('off')

plt.subplot(2, 3, 5)
plt.imshow(rotated_image, cmap='gray')
plt.title("Rotated Image (45°)")
plt.axis('off')

plt.show()
```

Output Images

1. Original Image

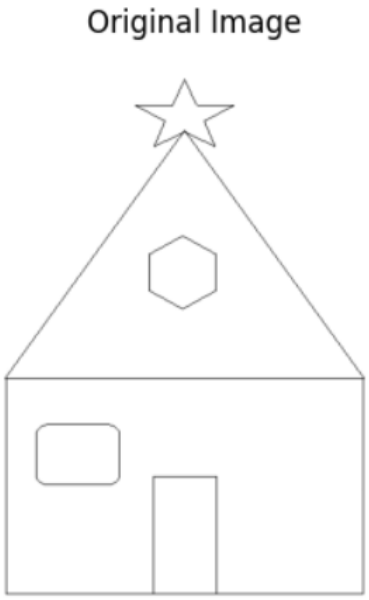


Figure 1: Grayscale input image loaded from file.

2. Line Detection using Canny and Hough Transform

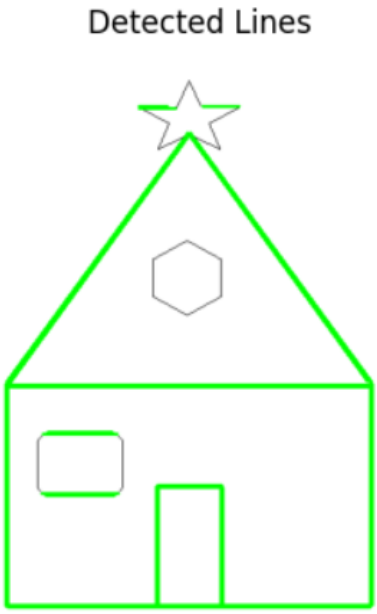


Figure 2: Detected lines overlaid on original image using Hough Transform.

3. Corner Detection using Harris

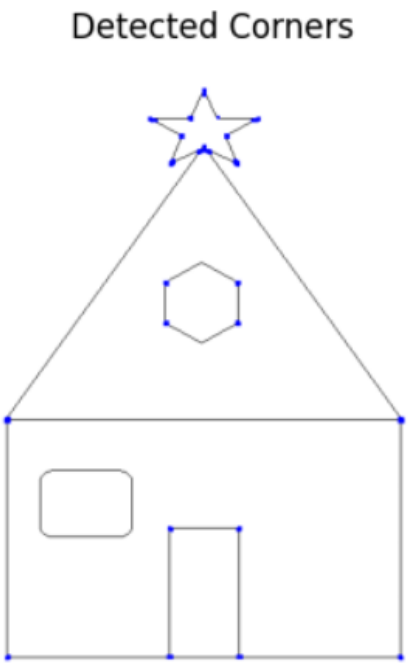


Figure 3: Detected corners highlighted in red using Harris Corner Detection.

4. SIFT Feature Detection

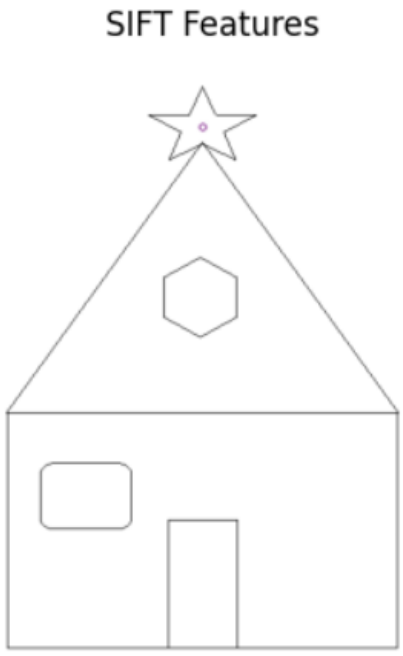


Figure 4: Keypoints detected and drawn using the SIFT algorithm.

5. Rotated Image

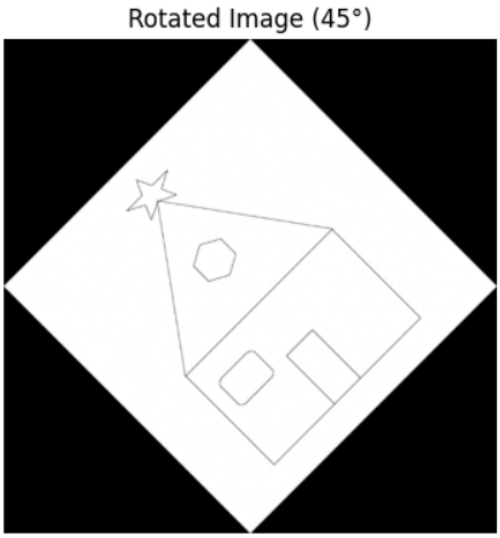


Figure 5: Image rotated by 45 degrees and resized to original dimensions.

Image 2

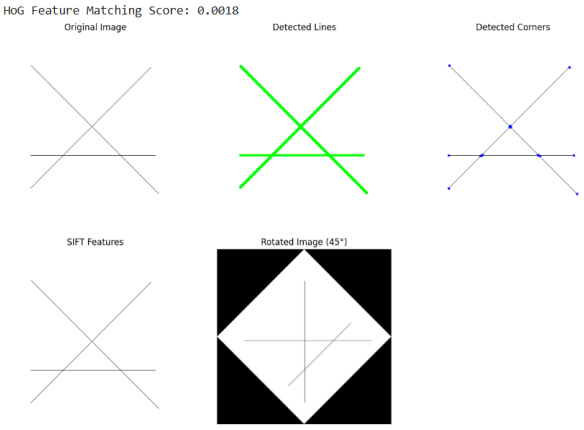


Image 3

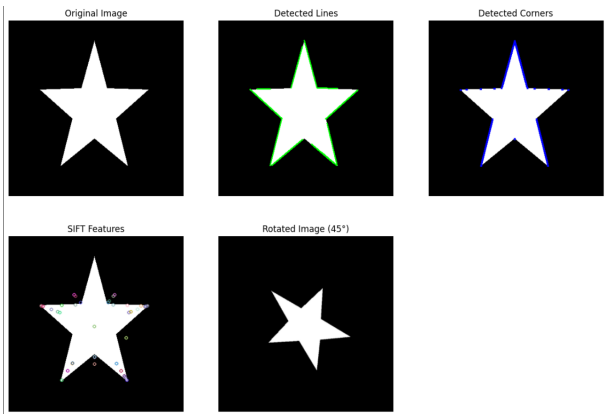


Image 4

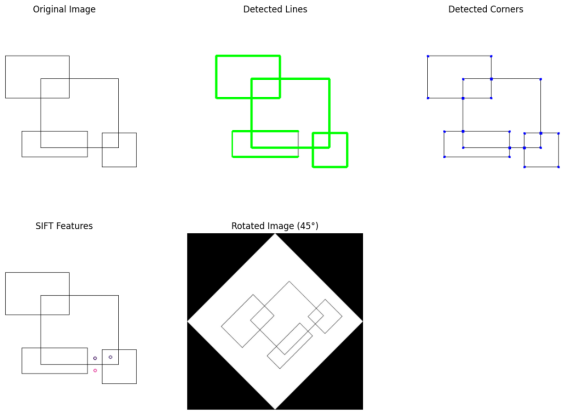
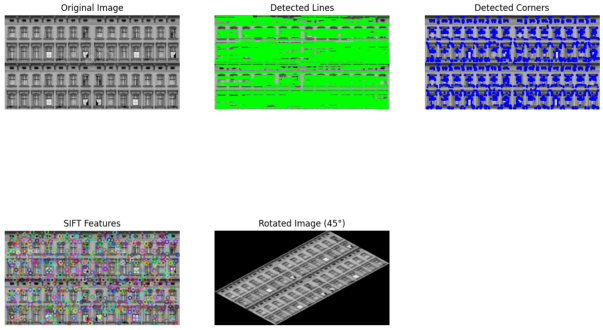


Image 5



6. HoG Feature Matching Score

Image 1 Matching Score: `0.1433`
Image 2 Matching Score: `0.0018`
Image 3 Matching Score: `0.0240`
Image 4 Matching Score: `0.0112`
Image 5 Matching Score: `0.4870`

Conclusion

This project demonstrates how to detect lines, corners, and features in images using various techniques in OpenCV and Skimage. Additionally, it shows how image similarity can be quantified using HoG descriptors and cosine similarity.