Computer Vision

Name: Abhishek Choudhary

Enrollment No: 04619051622

Lab - 4 : Harris Corner Detection

OBJECTIVE

To detect and visualize the corners (i.e., interest points or feature points) in an image using the Harris Corner Detection algorithm

Corner

Corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are important features in image generally termed as interest point

Measuring Uniqueness with SSD:

To quantify how much a small image window changes when shifted, we use the Sum of Squared Differences (SSD):

$$E(u,v) = \sum_{x,y} w(x,y)\,[I(x+u,y+v) - I(x,y)]^2$$

Where:

E(u,v): Change in intensity due to shift (u,v)

I: Pixel intensity

w(x,y): Weight (usually Gaussian)

We want to find regions where E(u, v) is large in all directions

We simplify the Equation using taylor series expansion and matrix M

Matrix M

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

R = det(M) - k * (trace(M))^2

det(M) = $\lambda_1 * \lambda_2$

trace(M) = $\lambda_1 + \lambda_2$

λ1 and λ2 are the eigenvalues of M. So the values of these eigenvalues decide whether a region is a corner, edge or flat. When |R| is small, which happens when λ1 and λ2 are small, the region is flat. When R<0, which happens when λ>>λ2 or vice versa, the region is an edge. When R is large, which happens when λ1 and λ2 are large and λ1∼λ2, the region is a corner.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import os
```

```
from google.colab import files

uploaded = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Blue chess board.jpg to Blue chess board (1).jpg
```

Reading and Displaying Image

```
img = cv2.imread("Blue chess board.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_gray = np.float32(img_gray)

plt.imshow(img_rgb)
plt.axis("off")
```

```
plt.title("Color Image")
plt.show()

plt.imshow(img_gray, cmap='gray')
plt.axis("off")
plt.title("Grayscale Image")
plt.show()
```

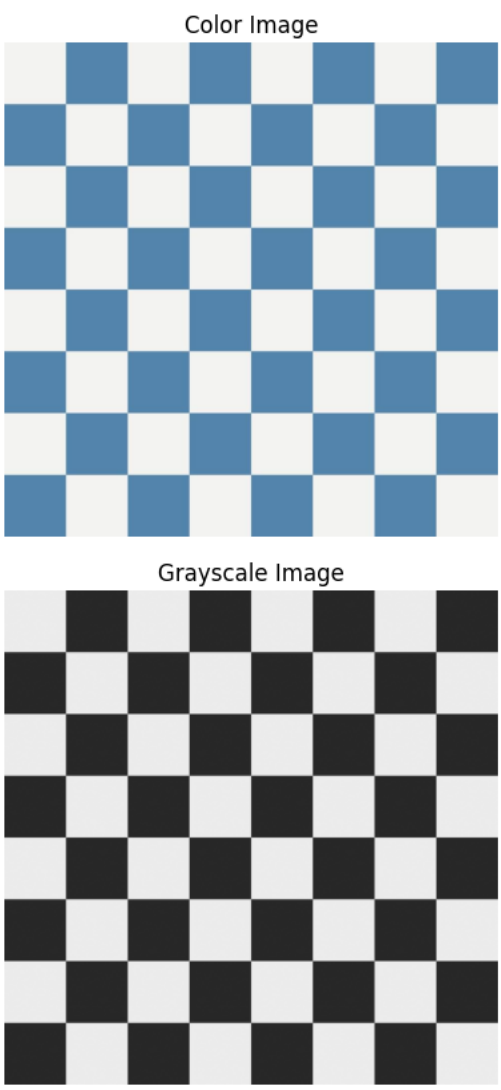


Image Gradients with Sobel Filters

To measure changes in intensity in x and y directions, i.e., gradients.

This tells us where the image is changing rapidly – possible edges or corners.

```
sobel_y=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
sobel_x=sobel_y.T
print("sobel_x")
print(sobel_x)

print("sobel_y")
print(sobel_y)
```

```
sobel_x
[[-1  0  1]
 [-2  0  2]
 [-1  0  1]]
sobel_y
[[-1 -2 -1]
 [ 0  0  0]
 [ 1  2  1]]
```

```
Ix=cv2.filter2D(img_gray,-1,sobel_x)
Iy=cv2.filter2D(img_gray,-1,sobel_y)
```

Computing Structure Tensor (Matrix M)

We apply Gaussian blur to smooth the values and suppress noise in local neighborhood.

```
Ixx=cv2.GaussianBlur(Ix*Ix,(3,3),1)
Iyy=cv2.GaussianBlur(Iy*Iy,(3,3),1)
Ixy=cv2.GaussianBlur(Ix*Iy,(3,3),1)
```

Harris Corner Detection

Harris Corner Detection is an algorithm used in computer vision to find corners (also called interest points) in an image.

We analyze a small window (usually 3×3 or 5×5) around each pixel.

1. In flat regions, there's no gradient change in any direction — shifting the window makes no difference.
2. Along an edge, changes are visible only in one direction — not distinctive.
3. In a corner, shifting the window in any direction causes significant change — highly distinctive

Harris Response

It is a score calculated for each pixel to decide if it's a corner, edge, or flat region. The three formulae utilized are -

1. R = detM-k*(traceM)
2. R = detM / traceM
3. R = λ1

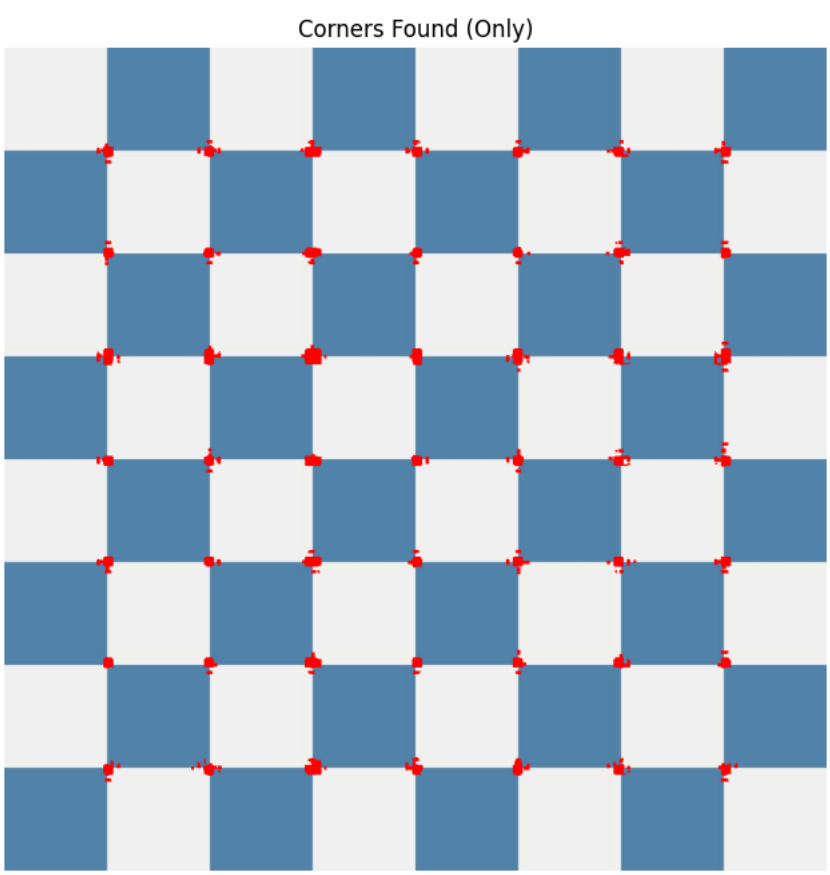
Function: harris_corner()

This function marks the corners detected in the image using the Harris response matrix by coloring them red.

Steps:

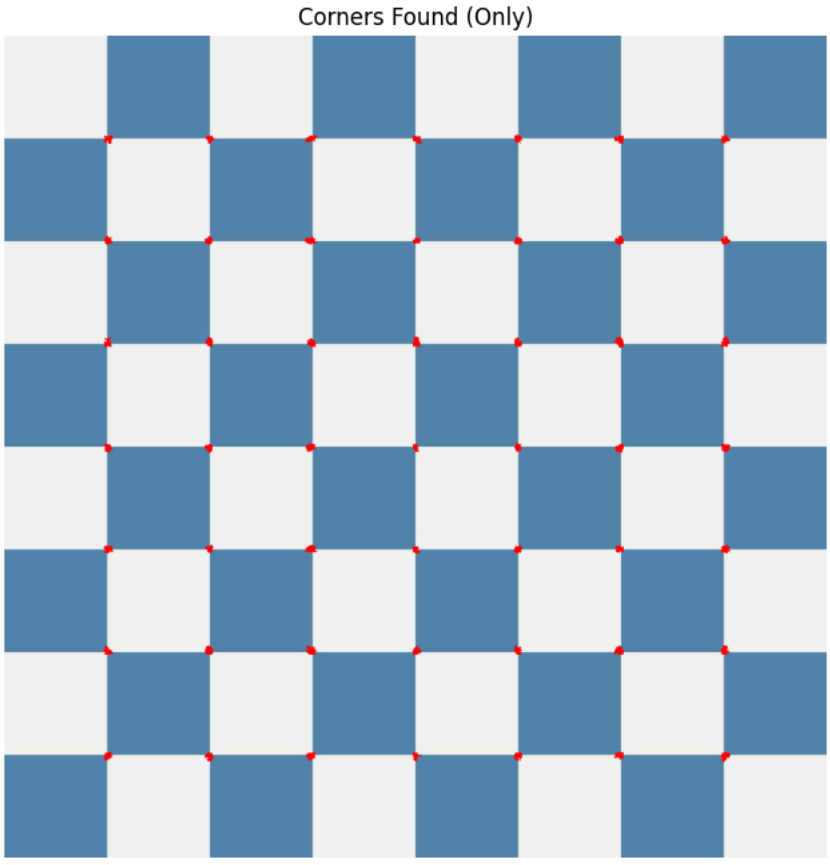
1. Create a copy of the image
2. Iterating over each pixel in the Harris response matrix. If the response r is greater than a threshold mark it red.
3. Display the result.

```
detM=Ixx*Iyy-(Ixy)**2
traceM=Ixx+Iyy
# Calculating lamda
sqrt_term = np.sqrt((traceM ** 2) - 4 * detM)
lambda1 = (traceM + sqrt_term) / 2
#lambda 2 = (traceM - sqrt_term) / 2


def harris_corner(image, harris_response):
    # Copy image for marking corners
    img_corners_only = np.copy(image)

    # Highlight only the corners
    for row_index, response in enumerate(harris_response):
        for col_index, r in enumerate(response):
            if r > 100:
                img_corners_only[row_index, col_index] = [255, 0, 0]  # Red

    # Plot
    plt.figure(figsize=(8, 8))
    plt.title("Corners Found (Only)")
    plt.imshow(img_corners_only)
    plt.axis("off")
    plt.show()


k=0.05
harris_response=detM-k*(traceM)
harris_corner(img_rgb, harris_response)
```

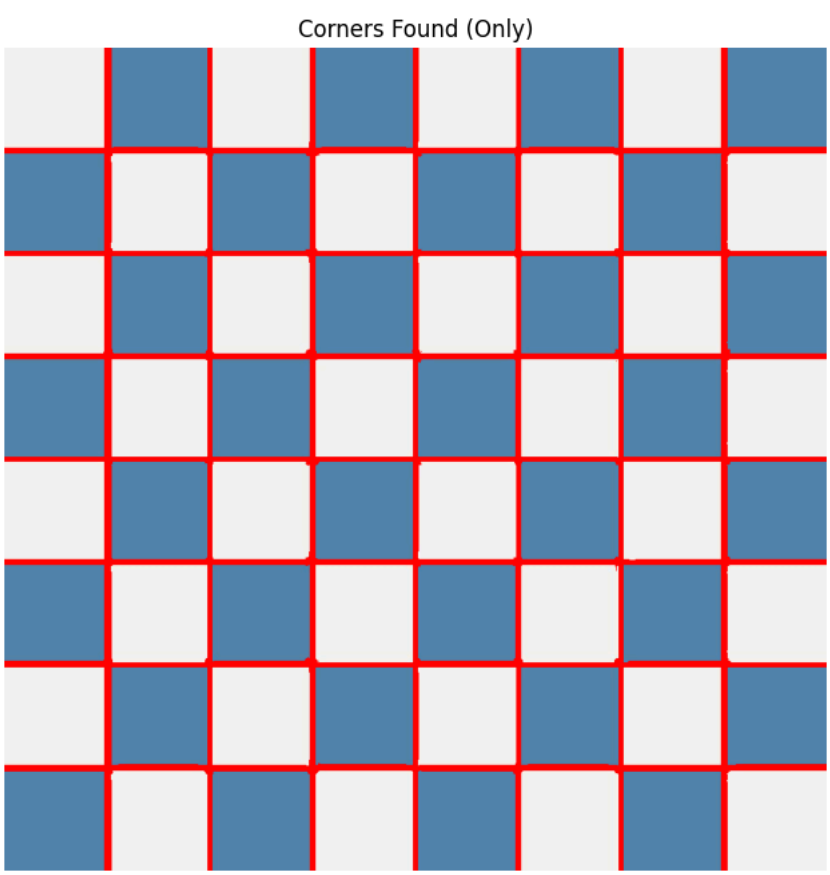



```
harris_response = detM / traceM
harris_corner(img_rgb, harris_response)
```

```
<ipython-input-78-0676d19bbeaa>:1: RuntimeWarning: invalid value encountered in divide
  harris_response = detM / traceM
```




```
harris_response= lambda1
harris_corner(img_rgb, harris_response)
```



Conclusion

Harris Corner Detection is a simple yet powerful technique to identify interest points in an image. With just a few lines of code, you can visualize and understand the structure hidden within images.