EXPT 6	Edge, Corner and Line Detection

AIM:-

To implement feature detection algorithms including Canny edge detector for robust edge detection, Harris corner detector for identifying corner points, and Hough transform for detecting straight lines in images.

CODE:-

```
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('input.jpg')

if img is None:
    print("Upload 'input.jpg' to Colab.")
else:
    # Create copies of the original image for displaying results
    img_canny = img.copy()
    img_harris = img.copy()
    img_hough = img.copy()
    img_blur = img.copy()
    img_threshold = img.copy()
    img_sobelx = img.copy()
    img_sobely = img.copy()


    # Canny Edge Detection
    gray_canny = cv2.cvtColor(img_canny, cv2.COLOR_BGR2GRAY)
    edges_canny = cv2.Canny(gray_canny, 100, 200)

    # Harris Corner Detection
```

```
    gray_harris = cv2.cvtColor(img_harris, cv2.COLOR_BGR2GRAY)
    gray_harris = np.float32(gray_harris)
    dst_harris = cv2.cornerHarris(gray_harris, 2, 3, 0.04)
    dst_harris = cv2.dilate(dst_harris, None)
    img_harris[dst_harris > 0.01 * dst_harris.max()] = [0, 0, 255]

    # Hough Line Detection
    gray_hough = cv2.cvtColor(img_hough, cv2.COLOR_BGR2GRAY)
    edges_hough = cv2.Canny(gray_hough, 50, 150, apertureSize=3)
    lines_hough = cv2.HoughLines(edges_hough, 1, np.pi / 180, 150)
    if lines_hough is not None:
        for rho, theta in lines_hough[:, 0]:
            a = np.cos(theta)
            b = np.sin(theta)
            x0 = a * rho
            y0 = b * rho
            x1 = int(x0 + 1000*(-b))
            y1 = int(y0 + 1000*(a))
            x2 = int(x0 - 1000*(-b))
            y2 = int(y0 - 1000*(a))
            cv2.line(img_hough, (x1, y1), (x2, y2), (0, 0, 255), 2)

    # Gaussian Blur
    img_blur = cv2.GaussianBlur(img_blur, (5, 5), 0)

    # Simple Thresholding
    gray_threshold = cv2.cvtColor(img_threshold, cv2.COLOR_BGR2GRAY)
    ret, img_threshold = cv2.threshold(gray_threshold, 127, 255,
cv2.THRESH_BINARY)

    # Sobel Edge Detection (X-direction)
    gray_sobelx = cv2.cvtColor(img_sobelx, cv2.COLOR_BGR2GRAY)
    sobelx = cv2.Sobel(gray_sobelx, cv2.CV_64F, 1, 0, ksize=5)
    abs_sobelx = np.absolute(sobelx)
```

```
img_sobelx = np.uint8(abs_sobelx)

# Sobel Edge Detection (Y-direction)
gray_sobely = cv2.cvtColor(img_sobely, cv2.COLOR_BGR2GRAY)
sobely = cv2.Sobel(gray_sobely, cv2.CV_64F, 0, 1, ksize=5)
abs_sobely = np.absolute(sobely)
img_sobely = np.uint8(abs_sobely)


# Display results in a 2x4 grid
fig, axes = plt.subplots(2, 4, figsize=(20, 10))

axes[0, 0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axes[0, 0].set_title('Original Image')
axes[0, 0].axis('off')

axes[0, 1].imshow(edges_canny, cmap='gray')
axes[0, 1].set_title('Canny Edges')
axes[0, 1].axis('off')

axes[0, 2].imshow(cv2.cvtColor(img_harris, cv2.COLOR_BGR2RGB))
axes[0, 2].set_title('Harris Corners')
axes[0, 2].axis('off')

axes[0, 3].imshow(cv2.cvtColor(img_hough, cv2.COLOR_BGR2RGB))
axes[0, 3].set_title('Hough Lines')
axes[0, 3].axis('off')

axes[1, 0].imshow(cv2.cvtColor(img_blur, cv2.COLOR_BGR2RGB))
axes[1, 0].set_title('Gaussian Blur')
axes[1, 0].axis('off')

axes[1, 1].imshow(img_threshold, cmap='gray')
axes[1, 1].set_title('Binary Threshold')
```

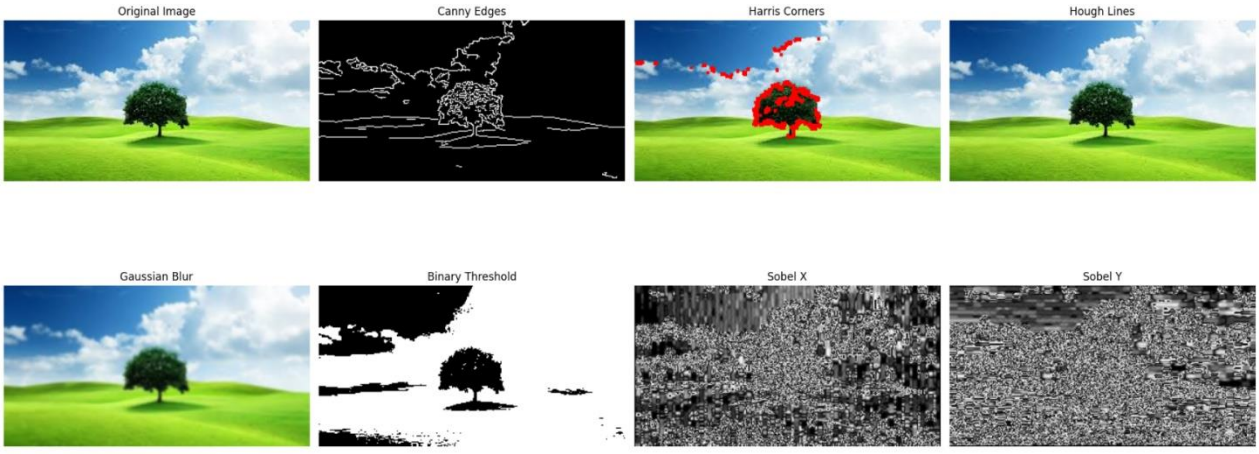
```
axes[1, 1].axis('off')

axes[1, 2].imshow(img_sobelx, cmap='gray')
axes[1, 2].set_title('Sobel X')
axes[1, 2].axis('off')

axes[1, 3].imshow(img_sobely, cmap='gray')
axes[1, 3].set_title('Sobel Y')
axes[1, 3].axis('off')


plt.tight_layout()
plt.show()
```

OUTPUT:-



RESULT:-

Successfully detected edges, corners, and lines in images using respective algorithms. Each method proved effective for extracting geometric features essential for object recognition, image segmentation, and scene understanding applications.