**Name-Akshat Sharma**

**Aim:-** *Corner Detection and Feature Detection (SIFT, SURF, CornerHarris)*

**Theory:-** Feature detection is a fundamental concept in computer vision, where the objective is to identify points of interest in an image that are robust and distinctive. These points, often referred to as features, provide useful information for image analysis, such as object recognition, image stitching, 3D reconstruction, and motion tracking.

Corner detection is a technique used to detect specific points in an image where the intensity changes sharply in multiple directions. These points are robust and can be tracked accurately between different views of the same scene.

**CODE:-**

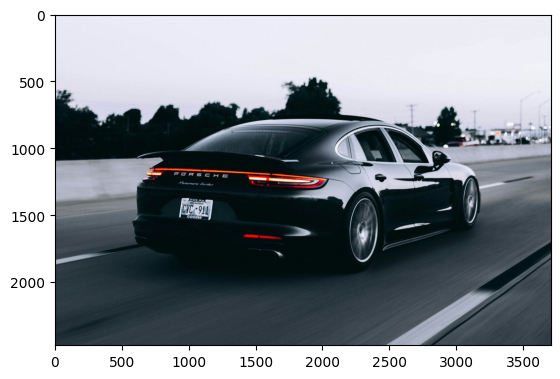
**1.Convert to RGB and Display Image**

**# Convert to RGB**

**image\_copy = np.copy(image)**

**image\_copy = cv2.cvtColor(image\_copy, cv2.COLOR\_BGR2RGB)**

**plt.imshow(image\_copy)**

****

**2. Harris Corner Detection**

**# Corner Harris Detection**

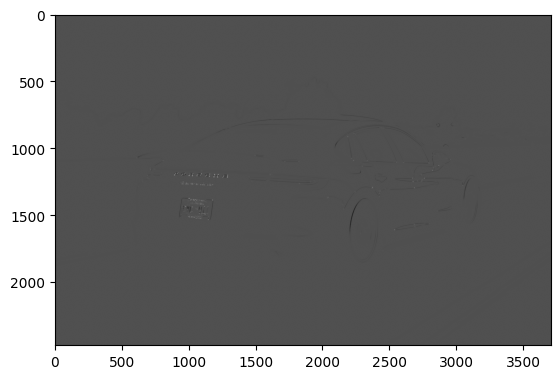
**gray = cv2.cvtColor(image\_copy, cv2.COLOR\_RGB2GRAY)**

**gray = np.float32(gray)**

**dst = cv2.cornerHarris(gray, 4, 7, 0.08)**

**dst = cv2.dilate(dst, None)**

**plt.imshow(dst, cmap='gray')**

****

**3. Detect and Display Corners**

**# Threshold and display detected corners**

**thresh = 0.01 \* dst.max()**

**corner\_image = np.copy(image\_copy)**

**for j in range(0, dst.shape[0]):**

**for i in range(0, dst.shape[1]):**

**if dst[j, i] > thresh:**

**cv2.circle(corner\_image, (i, j), 1, (0, 255, 0), 2)**

**plt.imshow(corner\_image)**

****

**4. Rotate the Image**

**# Rotate Image for Comparison**

**height, width = image.shape[:2]**

**angle = 45**

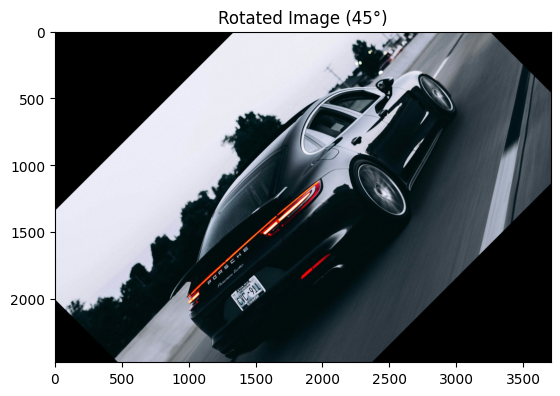
**rotation\_matrix = cv2.getRotationMatrix2D((width // 2, height // 2), angle, 1)**

**rotated\_image = cv2.warpAffine(image, rotation\_matrix, (width, height))**

**plt.imshow(cv2.cvtColor(rotated\_image, cv2.COLOR\_BGR2RGB))**

**plt.title(f'Rotated Image ({angle}°)')**

**plt.show()**

****

**5.**

**# Apply Corner Harris to rotated image**

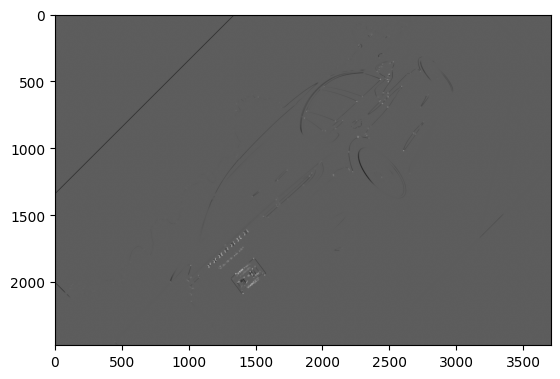
**gray = cv2.cvtColor(rotated\_image, cv2.COLOR\_BGR2GRAY)**

**gray = np.float32(gray)**

**dst\_rotated = cv2.cornerHarris(gray, 8, 9, 0.08)**

**dst\_rotated = cv2.dilate(dst\_rotated, None)**

**plt.imshow(dst\_rotated, cmap='gray')**

****

**6. Display Detected Corners on Rotated Image**

**# Threshold and display detected corners on rotated image**

**thresh = 0.01 \* dst\_rotated.max()**

**corner\_image\_rotated = np.copy(rotated\_image)**

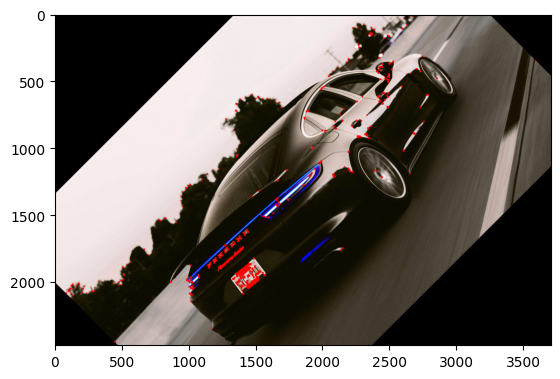
**for j in range(0, dst\_rotated.shape[0]):**

**for i in range(0, dst\_rotated.shape[1]):**

**if dst\_rotated[j, i] > thresh:**

**cv2.circle(corner\_image\_rotated, (i, j), 1, (255, 0, 0), 2)**

**plt.imshow(corner\_image\_rotated)**

****

**7. Compare Original and Rotated Image**

**# Compare original and rotated images**

**plt.figure(figsize=(10, 5))**

**plt.subplot(1, 2, 1)**

**plt.imshow(corner\_image)**

**plt.title("Original Image")**

**plt.axis("on")**

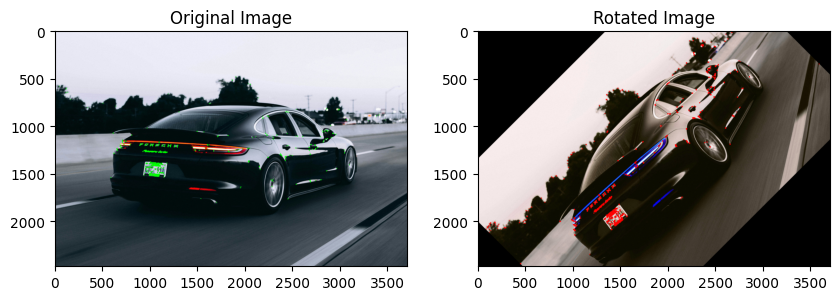
**plt.subplot(1, 2, 2)**

**plt.imshow(corner\_image\_rotated)**

**plt.title("Rotated Image")**

**plt.axis("on")**

**plt.show()**

****

**8. SIFT Feature Detection**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Load the image**

**image\_path = '/content/Car.jpg'**

**image = cv2.imread(image\_path)**

**# Check if image is loaded successfully**

**if image is None:**

**print("Error: Unable to load image!")**

**else:**

**# Convert image to RGB (OpenCV reads as BGR by default)**

**image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# Convert image to grayscale**

**gray = cv2.cvtColor(image\_rgb, cv2.COLOR\_RGB2GRAY)**

**# SIFT Feature Detection**

**sift = cv2.SIFT\_create()**

**keypoints\_sift = sift.detect(gray, None)**

**# Draw the keypoints on the image**

**sift\_image = cv2.drawKeypoints(image\_rgb, keypoints\_sift, None, (0,255,0), flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)**

**# Show the image with keypoints**

**plt.figure(figsize=(10, 10)) # Make the image bigger**

**plt.imshow(sift\_image)**

**plt.title("SIFT Keypoints")**

**plt.axis("off")**

**plt.show()**

****

**9. SURF Feature Detection**

**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Load the image**

**image\_path = 'Car.jpg'**

**image = cv2.imread(image\_path)**

**# Check if image is loaded successfully**

**if image is None:**

**print("Error: Unable to load image!")**

**else:**

**# Convert image to RGB (OpenCV reads as BGR by default)**

**image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# Convert image to grayscale**

**gray = cv2.cvtColor(image\_rgb, cv2.COLOR\_RGB2GRAY)**

**# ORB Feature Detection (Fallback for SURF)**

**orb = cv2.ORB\_create()**

**keypoints\_orb = orb.detect(gray, None)**

**# Draw the keypoints on the image**

**orb\_image = cv2.drawKeypoints(image\_rgb, keypoints\_orb, None, (0,255,0), flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)**

**# Show the image with keypoints**

**plt.figure(figsize=(10, 10)) # Make the image bigger**

**plt.imshow(orb\_image)**

**plt.title("ORB Keypoints")**

**plt.axis("off")**

**plt.show()**

****

**CONCLUSION:-**

Feature detection plays a critical role in many computer vision applications. The Corner Harris Detector is a simple, efficient corner detection method but lacks robustness to scale and rotation changes. SIFT, despite being computationally expensive, remains one of the most powerful algorithms due to its high accuracy and invariance properties. SURF offers a compromise between speed and accuracy, making it suitable for real-time applications.

The choice of algorithm depends largely on the application requirements, such as the need for invariance to scale, rotation, or robustness to noise. Understanding the strengths and limitations of each method is essential for selecting the most appropriate technique for a particular task.

Corner Harris: The Harris corner detection method is effective in detecting corners in both original and rotated images. However, rotation can cause slight misalignment of corners in transformed images.

SIFT vs SURF: SIFT is more accurate but slower, making it suitable for precise applications. SURF, while slightly less accurate, is faster and can be used in real-time scenarios. Both methods detect keypoints that are invariant to scale and rotation.

Comparison: Harris corner detection is good for detecting corner points, while SIFT and SURF are better for identifying distinctive feature points. Depending on the application, either could be preferred based on the need for speed (SURF) or accuracy (SIFT).