**ASSIGNMENT – PROGRAM 6**

**KEY POINT DETECTION COMPARISON**

**AIM:**

To implement and compare four keypoint detection algorithms, Scale-Invariant Feature Transform (SIFT), Shi-Tomasi Corner Detector, Speeded-Up Robust Features (SURF), Good Features to Track (GFTT) and Harris Corner Detector.

**ALGORITHM:**

1. Load and Preprocess the Image

1.1 Load the image using cv2.imread(). 1.2 Convert the image to grayscale using cv2.cvtColor().

2. SIFT Keypoint Detection

2.1 Initialize the SIFT detector using cv2.SIFT\_create(). 2.2 Record the start time using time.time(). 2.3 Detect keypoints and compute descriptors with sift.detectAndCompute(gray, None). 2.4 Calculate the computation time by subtracting the start time from the current time. 2.5 Print the number of keypoints detected and the computation time.

3. Visualize SIFT Keypoints

3.1 Draw the detected keypoints on the image using cv2.drawKeypoints(). 3.2 Display the image with keypoints using plt.imshow().

4. Harris Corner Detection

4.1 Convert the grayscale image to float32 using np.float32(). 4.2 Record the start time using time.time(). 4.3 Apply the Harris Corner Detector using cv2.cornerHarris(). 4.4 Dilate the corner response image to enhance corner points using cv2.dilate(). 4.5 Identify keypoints by thresholding the response map. 4.6 Calculate the computation time by subtracting the start time from the current time. 4.7 Print the number of keypoints detected and the computation time.

5. Visualize Harris Keypoints

5.1 Draw circles around the detected corners on the original image. 5.2 Display the image with detected corners using plt.imshow().

6. Shi-Tomasi Corner Detection

6.1 Record the start time using time.time(). 6.2 Apply the Shi-Tomasi Corner Detector using cv2.goodFeaturesToTrack(). 6.3 Convert the corner coordinates to integer values. 6.4 Calculate the computation time by subtracting the start time from the current time. 6.5 Print the number of keypoints detected and the computation time.

7. Visualize Shi-Tomasi Keypoints

7.1 Draw circles around the detected corners on the original image. 7.2 Display the image with detected corners using plt.imshow().

8. GFTT Keypoint Detection

8.1 Record the start time using time.time(). 8.2 Apply the GFTT Detector using cv2.goodFeaturesToTrack(). 8.3 Convert the corner coordinates to integer values. 8.4 Calculate the computation time by subtracting the start time from the current time. 8.5 Print the number of keypoints detected and the computation time.

9. Visualize GFTT Keypoints

9.1 Draw circles around the detected corners on the original image. 9.2 Display the image with detected corners using plt.imshow().

**CODE:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

import time

def harris\_corner\_detection(img):

    start\_time = time.time()

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    gray = np.float32(gray)

    dst = cv2.cornerHarris(gray, 2, 3, 0.04)

    dst = cv2.dilate(dst, None)

    img[dst > 0.01 \* dst.max()] = [0, 0, 255]

    end\_time = time.time()

    return img, end\_time - start\_time

def shi\_tomasi\_corner\_detection(img):

    start\_time = time.time()

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    corners = cv2.goodFeaturesToTrack(gray, 25, 0.01, 10)

    corners = np.int0(corners)

    for i in corners:

        x, y = i.ravel()

        cv2.circle(img, (x, y), 3, 255, -1)

    end\_time = time.time()

    return img, end\_time - start\_time

def sift\_keypoint\_detection(img):

    start\_time = time.time()

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    sift = cv2.SIFT\_create()

    keypoints, descriptors = sift.detectAndCompute(gray, None)

    img = cv2.drawKeypoints(img, keypoints, None, flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

    end\_time = time.time()

    return img, end\_time - start\_time

def gftt\_keypoint\_detection(img):

    start\_time = time.time()

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    corners = cv2.goodFeaturesToTrack(gray, 100, 0.01, 10)

    corners = np.int0(corners)

    for corner in corners:

        x, y = corner.ravel()

        cv2.circle(img, (x, y), 3, 255, -1)

    end\_time = time.time()

    return img, end\_time - start\_time

def perform\_keypoint\_detection(image\_path):

    img = cv2.imread(image\_path)

    if img is None:

        print(f"Could not open or find the image: {image\_path}")

        return

    plt.figure(figsize=(15, 10))

    img\_harris, time\_harris = harris\_corner\_detection(img.copy())

    plt.subplot(231)

    plt.imshow(cv2.cvtColor(img\_harris, cv2.COLOR\_BGR2RGB))

    plt.title(f'Harris Corner Detection\nTime: {time\_harris:.4f}s')

    img\_shi\_tomasi, time\_shi\_tomasi = shi\_tomasi\_corner\_detection(img.copy())

    plt.subplot(232)

    plt.imshow(cv2.cvtColor(img\_shi\_tomasi, cv2.COLOR\_BGR2RGB))

    plt.title(f'Shi-Tomasi Corner Detection\nTime: {time\_shi\_tomasi:.4f}s')

    img\_sift, time\_sift = sift\_keypoint\_detection(img.copy())

    plt.subplot(233)

    plt.imshow(cv2.cvtColor(img\_sift, cv2.COLOR\_BGR2RGB))

    plt.title(f'SIFT Keypoint Detection\nTime: {time\_sift:.4f}s')

    img\_gftt, time\_gftt = gftt\_keypoint\_detection(img.copy())

    plt.subplot(234)

    plt.imshow(cv2.cvtColor(img\_gftt, cv2.COLOR\_BGR2RGB))

    plt.title(f'GFTT Detection\nTime: {time\_gftt:.4f}s')

    plt.tight\_layout()

    plt.show()

# Replace 'path/to/your/image.jpg' with the actual path to your image

perform\_keypoint\_detection(r"D:\\CV\bg.jpg")

**OUTPUT:**







