**CV Practical No.: 7**

**Aim: Shift Invariant Fourier Transform**

**New Concept:**

**i. cv2.SIFT\_create():** it is a function in OpenCV (a popular computer vision library) that creates a SIFT detector object. This object can be used to detect key points (distinct features) in an image and compute descriptors (representations) for those key points.

**ii. sift.detectAndCompute:** it is a method used to detect key points and compute descriptors for an image. Points in the image that have distinctive patterns and can be used to identify parts of the image (e.g., corners, edges). A vector representation for each key point, capturing the local image patterns around the key point.

**iii. cv2.BFMatcher():** it is a Brute Force Matcher in OpenCV. It is used to match feature descriptors between two images based on their distance. A brute force approach means it compares every descriptor from one set to every descriptor from another set to find the best match.

**iv. bf.knnMatch:** it is a method of the Brute Force Matcher (BFMatcher) used for finding the k-nearest neighbors between two sets of descriptors. K-Nearest Neighbors (KNN) means that for each descriptor from the first image, it will find k closest matches from the second image.

**v. cv2.drawMatches:** it is a function that visually draws the matches between key points in two images. It helps to visualize the feature matching process, showing how key points from one image are aligned with key points in another image.

**vi. cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS:** this is a flag used with cv2.drawMatches to control how key points are drawn. cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS means that key points that do not have matches will not be drawn. If you want to show only the matches (and not the single, unmatched key points), you can use this flag. If you want to draw all key points (even those without matches), you can use a different flag (e.g., cv2.DrawMatchesFlags\_DEFAULT).

**Theory:**

**Program:**

import cv2

# Load two images

image1 = cv2.imread("scenery\_1.jpg")

image2 = cv2.imread("scenery\_2.jpg")

# Convert images to grayscale

gray1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2GRAY)

gray2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2GRAY)

# Initialize SIFT detector

sift = cv2.SIFT\_create()

# Detect keypoints and compute descriptors for both images

keypoints1, descriptors1 = sift.detectAndCompute(gray1, None)

keypoints2, descriptors2 = sift.detectAndCompute(gray2, None)

# Initialize a Brute Force Matcher

bf = cv2.BFMatcher()

# Match descriptors between the two images

matches = bf.knnMatch(descriptors1, descriptors2, k=2)

# Apply ratio test to filter good matches

good\_matches = []

for m, n in matches:

if m.distance < 0.5 \* n.distance:

good\_matches.append(m)

# Draw matches

matched\_image = cv2.drawMatches(image1, keypoints1, image2, keypoints2, good\_matches, None, flags=cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)

# Display the matched image

cv2.imshow('Key Point Matches', matched\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output:**

****