**Finger-print matching code**

**Done by:**

**Lakshmikanthreddy Guddila**

**Moulish G A**

**Code:**

import os

import cv2

# Read the sample altered fingerprint image

sample = cv2.imread("SOCOFing/Altered/Altered-hard/150\_\_M\_Right\_index\_finger\_Obl.BMP")

# Initialize variables to track the best match

best\_score = 0

filename = None

image = None

kp1, kp2, mp = None, None, None

# Iterate over the real fingerprint images

for file in os.listdir("SOCOFing/Real")[:1000]:

    fingerprint\_image = cv2.imread("SOCOFing/Real/" + file)

    sift = cv2.SIFT\_create()

    # Detect and compute keypoints and descriptors

    keypoints\_1, descriptors\_1 = sift.detectAndCompute(sample, None)

    keypoints\_2, descriptors\_2 = sift.detectAndCompute(fingerprint\_image, None)

    # Match descriptors using FLANN-based matcher

    matches = cv2.FlannBasedMatcher({'algorithm': 1, 'trees': 10}, {}).knnMatch(descriptors\_1, descriptors\_2, k=2)

    # Apply ratio test to find good matches

    match\_points = []

    for p, q in matches:

        if p.distance < 0.75 \* q.distance:  # Adjusted ratio test threshold

            match\_points.append(p)

    # Determine the number of keypoints

    keypoints = min(len(keypoints\_1), len(keypoints\_2))

    # Update best score and corresponding image if the current match is better

    if len(match\_points) / keypoints \* 100 > best\_score:

        best\_score = len(match\_points) / keypoints \* 100

        filename = file

        image = fingerprint\_image

        kp1, kp2, mp = keypoints\_1, keypoints\_2, match\_points

# Print the best match and score

print("BestMatch : " + filename)

print("Score : " + str(best\_score))

# Draw matches on the best match image

result = cv2.drawMatches(sample, kp1, image, kp2, mp, None)

result = cv2.resize(result, None, fx=4, fy=4)

# Display the result

cv2.imshow("Result", result)

cv2.waitKey(0)

cv2.destroyAllWindows()