“ Fingerprint – Recognition – with – SIFT - ALGORITHM ”

Authors
- Adithya Harsha-
- DePaul University -
- Software Presentation -

What is SIFT (Scale-Invariant Feature Transform)

- SIFT is a computer vision algorithm used to detect and describe local features in images. It was developed by David Lowe in 1999 and is widely used for object recognition, including biometric applications like fingerprint matching.
- Key Property of SIFT
- Scale and Rotation Invariance:
  - Detects features regardless of image size, scale, or rotation
  - Handles variations due to **rotation, pressure, distortion**
- Purpose of SIFT
- Identify Unique Keypoints:
  - Extracts **distinct local features** from fingerprint images
  - Matches keypoints between **database and test samples**
  - Robust to **altered or distorted fingerprints**

Why SIFT is Suitable for Fingerprint Recognition

- **Distinctive Local Features:**
- Fingerprints have unique ridge patterns and minutiae points (ridge endings, bifurcations).
- SIFT identifies these features, enabling precise fingerprint matching.

- **Scale and Rotation Invariance:**
- SIFT is robust to changes in scale and rotation,
- ensuring accurate feature detection even if the fingerprint is rotated or resized.

- **Altered Fingerprints:**
- Even when fingerprints are altered (obliterations, rotations, z-cuts),
- SIFT can detect unaltered sections and match keypoints between original and altered fingerprints.

How SIFT Works in Fingerprint Recognition

- **Keypoint Detection:**
  - Identifies distinct points (e.g., ridges, corners) in both the sample and database fingerprints.
- **Difference of Gaussian (DoG):**
  - Detects keypoints by identifying areas that stand out (e.g., edges, corners).
- **Scale Space:**
  - Creates scaled versions of the image to detect keypoints at various scales.
- **Descriptor Calculation:**
  - Computes descriptors (feature vectors) for each keypoint based on local gradients.
- **Matching Keypoints:**
  - Matches descriptors from both fingerprints using a nearest-neighbor algorithm.
- **Lowe’s Ratio Test:**
  - Filters out poor matches, keeping only high-quality keypoint matches.
- **Match Score:**
  - Calculates a match score based on the number of matching keypoints.

SOCOFing Dataset Overview

•**Subjects & Fingerprints**:
•600 unique subjects, each with 10 fingerprints (both hands), totaling ~6,000 images.

•**Image Types**:
•**Real**: 6,000 unaltered images.
•**Altered**: 6,000 tampered images (split between Hard and Easy alterations).

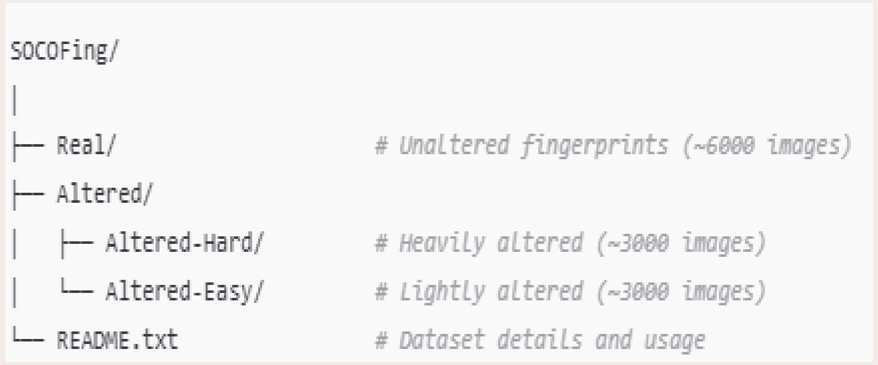
•**File Format**:
•All images are in high-quality BMP format.

•**Naming Convention**:
•Files named as <SubjectID>__<Gender>_<Finger>_<Alteration>.BMP, indicating the subject, finger, and type of alteration.

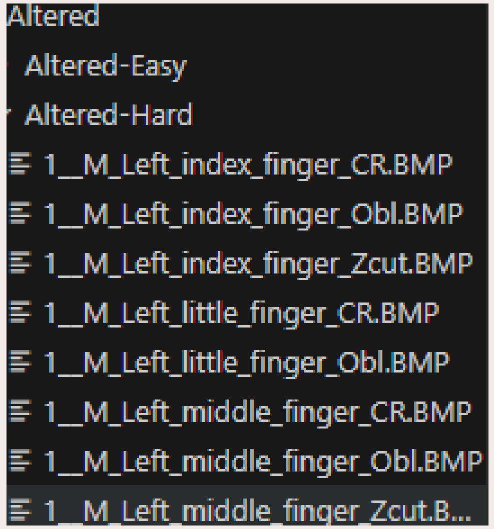
•**Alteration Types**:
•**Altered-Easy**: Light changes.
•**Altered-Hard**: Significant modifications.

•**Applications in Research**:
•Useful for testing fingerprint recognition accuracy, tampering detection, and anti-spoofing systems.

•**Dataset Size**:
•Over 12,000 fingerprint images in total (Real + Altered).



```
SOCOFing/
│
├── Real/                     # Unaltered fingerprints (~6000 images)
├── Altered/
│   ├── Altered-Hard/         # Heavily altered (~3000 images)
│   └── Altered-Easy/         # Lightly altered (~3000 images)
└── README.txt                # Dataset details and usage
```

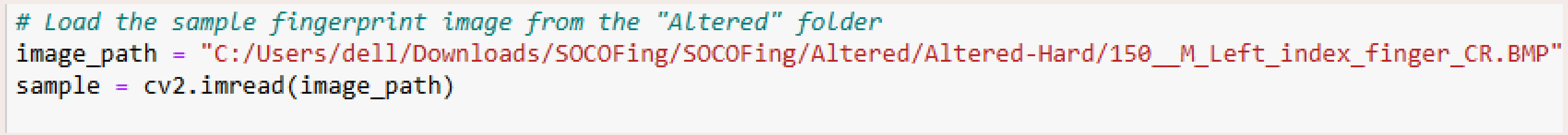


```
Altered
  Altered-Easy
  Altered-Hard
    1__M_Left_index_finger_CR.BMP
    1__M_Left_index_finger_Obl.BMP
    1__M_Left_index_finger_Zcut.BMP
    1__M_Left_little_finger_CR.BMP
    1__M_Left_little_finger_Obl.BMP
    1__M_Left_middle_finger_CR.BMP
    1__M_Left_middle_finger_Obl.BMP
    1__M_Left_middle_finger_Zcut.B...
```

How SIFT helps in Fingerprint Matching

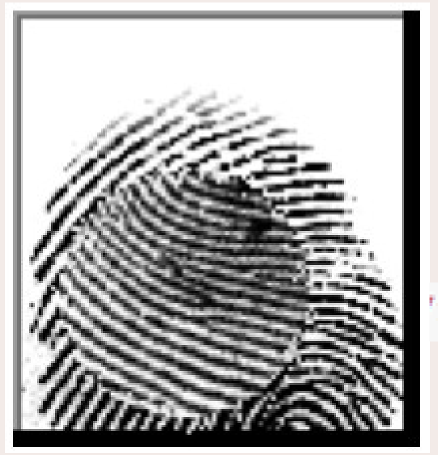
1. Load the dataset with necessary library

2. Load the sample fingerprint image from the "Altered" folder



```
# Load the sample fingerprint image from the "Altered" folder
image_path = "C:/Users/dell/Downloads/SOCOFing/SOCOFing/Altered/Altered-Hard/150__M_Left_index_finger_CR.BMP"
sample = cv2.imread(image_path)
```

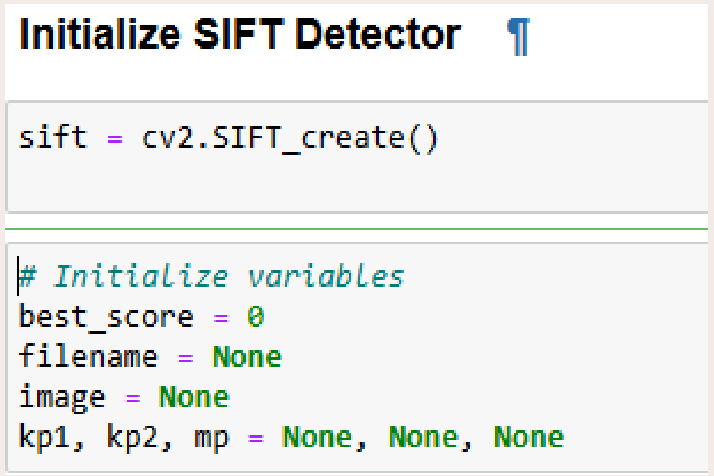
3. Display the Sample Finger print Image



```
'150__M_Left_index_finger_CR.BMP'
```

4. Initialize SIFT Detector

The Scale-Invariant Feature Transform (SIFT) detector is initialized to detect keypoints and descriptors in the fingerprint images. It's widely used in feature matching tasks.

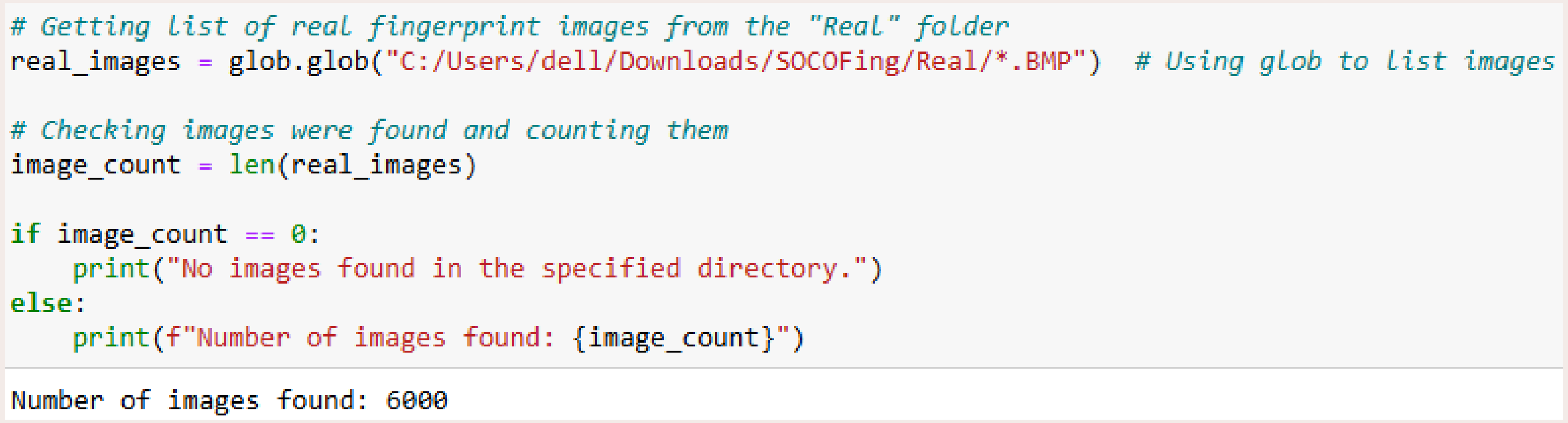


Initialize SIFT Detector ¶

```
sift = cv2.SIFT_create()
```

```
# Initialize variables
best_score = 0
filename = None
image = None
kp1, kp2, mp = None, None, None
```

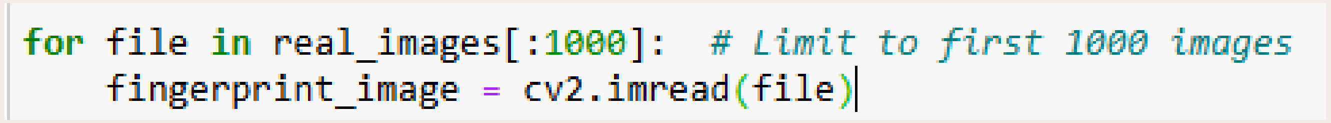
5. Load Real Fingerprint Images



```
# Getting list of real fingerprint images from the "Real" folder
real_images = glob.glob("C:/Users/dell/Downloads/SOCOFing/Real/*.BMP")  # Using glob to list images

# Checking images were found and counting them
image_count = len(real_images)

if image_count == 0:
    print("No images found in the specified directory.")
else:
    print(f"Number of images found: {image_count}")
```

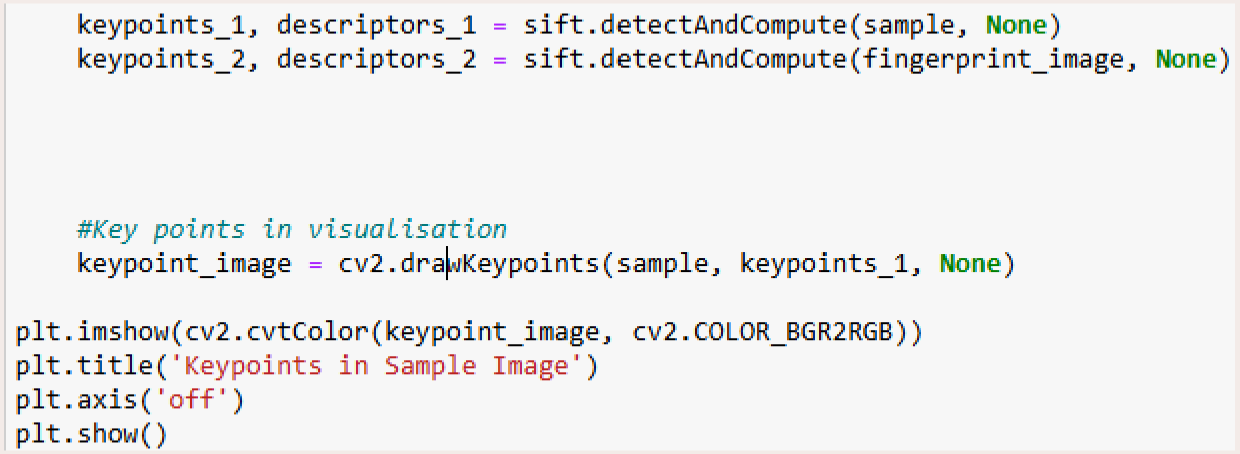
```
Number of images found: 6000
```

6. Iterates over The code iterates through the real fingerprint images, limiting it to the first 1000 images for efficiency. For each image, it loads the fingerprint for comparison with the sample.

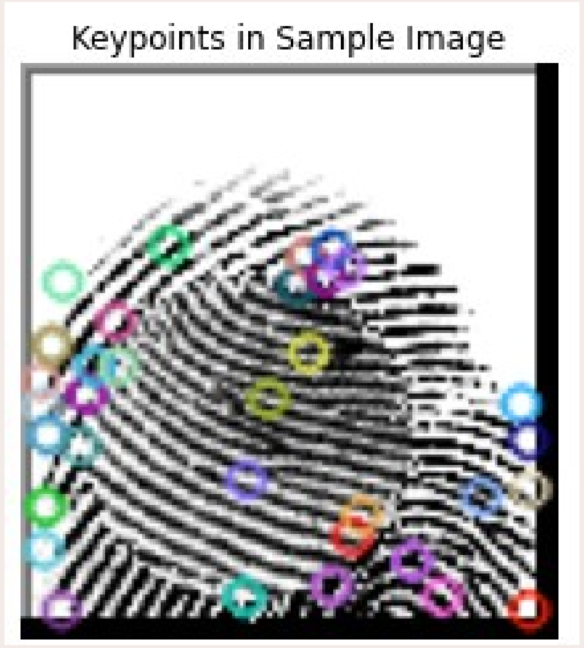


```
for file in real_images[:1000]:  # Limit to first 1000 images
    fingerprint_image = cv2.imread(file)
```

7. Compute Keypoints and Descriptors

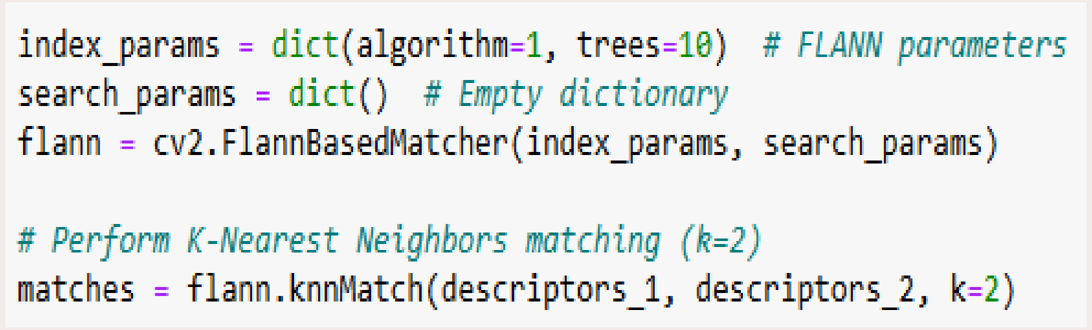


```
    keypoints_1, descriptors_1 = sift.detectAndCompute(sample, None)
    keypoints_2, descriptors_2 = sift.detectAndCompute(fingerprint_image, None)




    #Key points in visualisation
    keypoint_image = cv2.drawKeypoints(sample, keypoints_1, None)

plt.imshow(cv2.cvtColor(keypoint_image, cv2.COLOR_BGR2RGB))
plt.title('Keypoints in Sample Image')
plt.axis('off')
plt.show()
```



8. Perform Feature Matching with FLANN

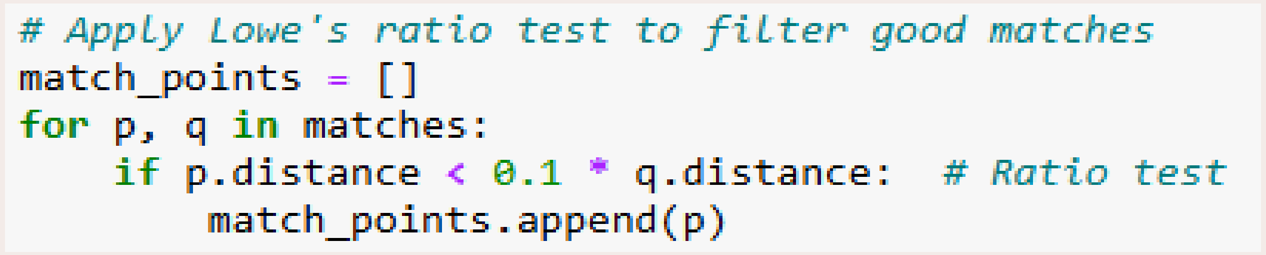
we use the **FLANN-based matcher** to find the best match between the keypoints of the two images. It performs **K-Nearest Neighbors (k=2)** matching.



```
index_params = dict(algorithm=1, trees=10)  # FLANN parameters
search_params = dict()  # Empty dictionary
flann = cv2.FlannBasedMatcher(index_params, search_params)

# Perform K-Nearest Neighbors matching (k=2)
matches = flann.knnMatch(descriptors_1, descriptors_2, k=2)
```

9. Apply Lowe's Ratio Test

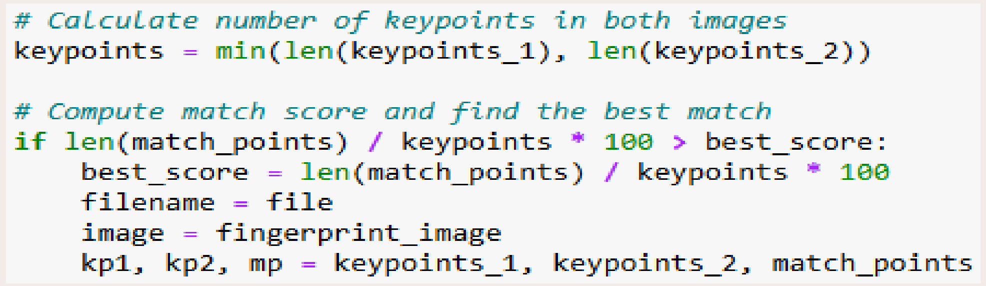
- This implements **Lowe's Ratio Test** to filter out weak matches. Only the good matches, where the nearest match is much closer than the second-closest match, are retained.



```
# Apply Lowe's ratio test to filter good matches
match_points = []
for p, q in matches:
    if p.distance < 0.1 * q.distance:  # Ratio test
        match_points.append(p)
```

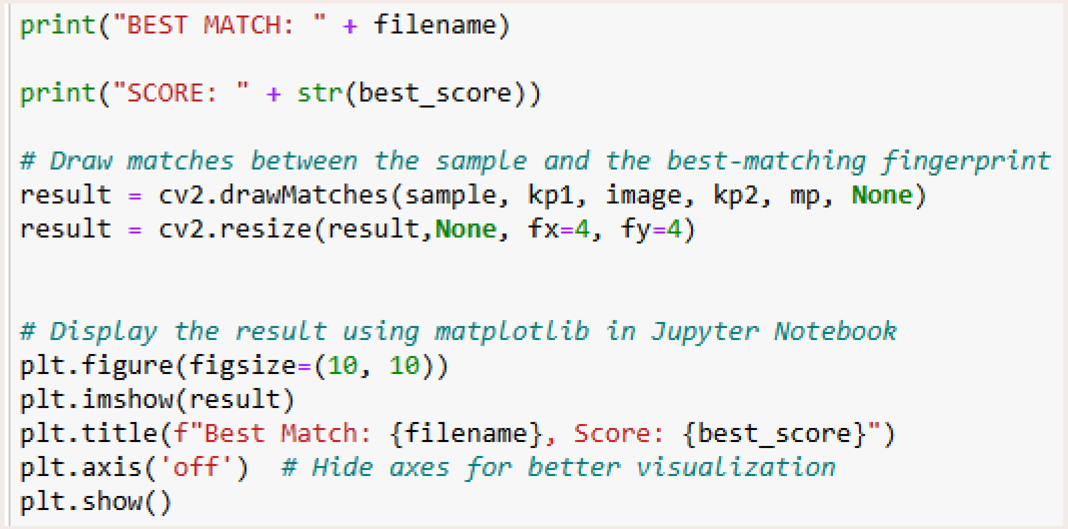
12. Calculating Match Score

- Matching score is calculated by dividing the number of good matches by the total keypoints. If the current score is better than the previous best, the score is updated, and the best-matching fingerprint is saved.



```
# Calculate number of keypoints in both images
keypoints = min(len(keypoints_1), len(keypoints_2))

# Compute match score and find the best match
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
```

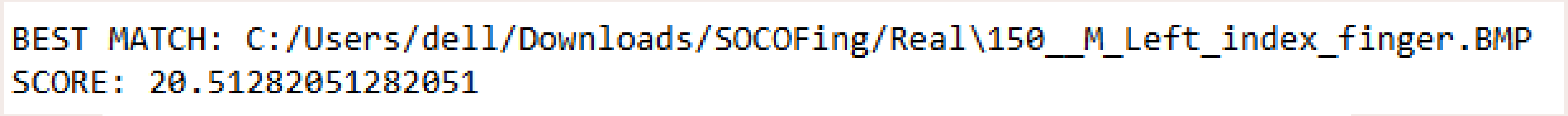
11.Best Match Identification and Visualization of Fingerprint Matching



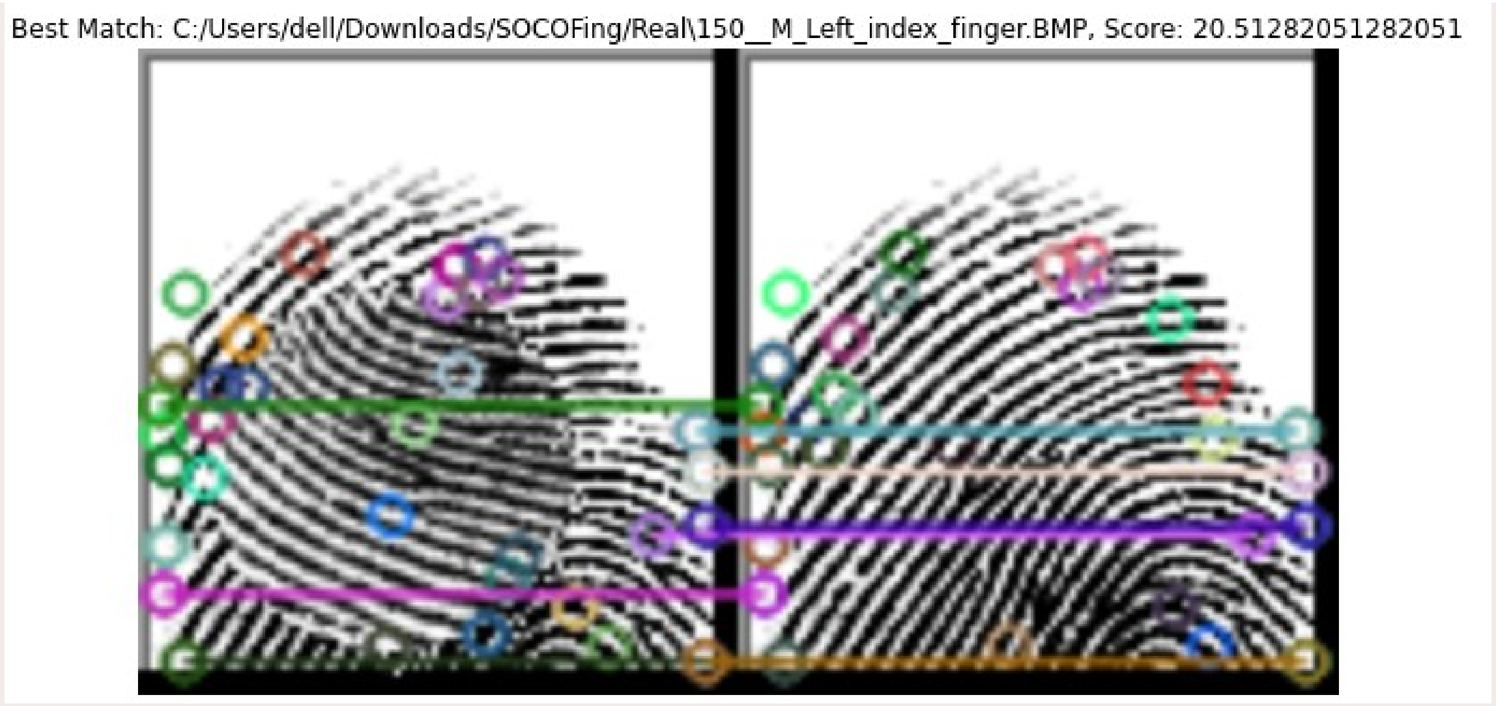
```
print("BEST MATCH: " + filename)

print("SCORE: " + str(best_score))

# Draw matches between the sample and the best-matching fingerprint
result = cv2.drawMatches(sample, kp1, image, kp2, mp, None)
result = cv2.resize(result,None, fx=4, fy=4)


# Display the result using matplotlib in Jupyter Notebook
plt.figure(figsize=(10, 10))
plt.imshow(result)
plt.title(f"Best Match: {filename}, Score: {best_score}")
plt.axis('off')  # Hide axes for better visualization
plt.show()
```

Final Summary: Visuals & Best Score

- The fingerprint comparison from our **dataset** yielded best match, confirming synchronization between the sample and the altered image, despite modifications.



```
BEST MATCH: C:/Users/dell/Downloads/SOCOFing/Real\150__M_Left_index_finger.BMP
SCORE: 20.51282051282051
```



Best Match: C:/Users/dell/Downloads/SOCOFing/Real\150__M_Left_index_finger.BMP, Score: 20.51282051282051

Thank you

Adithya Harsha

Aharsha@depaul.edu

Data Science Capstone - Software Presentation

NER with BERT