SAPIENZA
UNIVERSITÀ DI ROMA

Project presentation
Contactless Fingerprint Recognition

"Biometric Systems" course a.y. 2022/23

Professor: Maria De Marsico

Candidates:
- Clizia Giorgia Manganaro, manganaro.2017897@studenti.uniroma1.it
- Alessio Palma, palma.1837493@studenti.uniroma1.it
- Davide Santoro, santoro.1843664@studenti.uniroma1.it
- Gianluca Vannoli, vannoli.1837649@studenti.uniroma1.it

Introduction

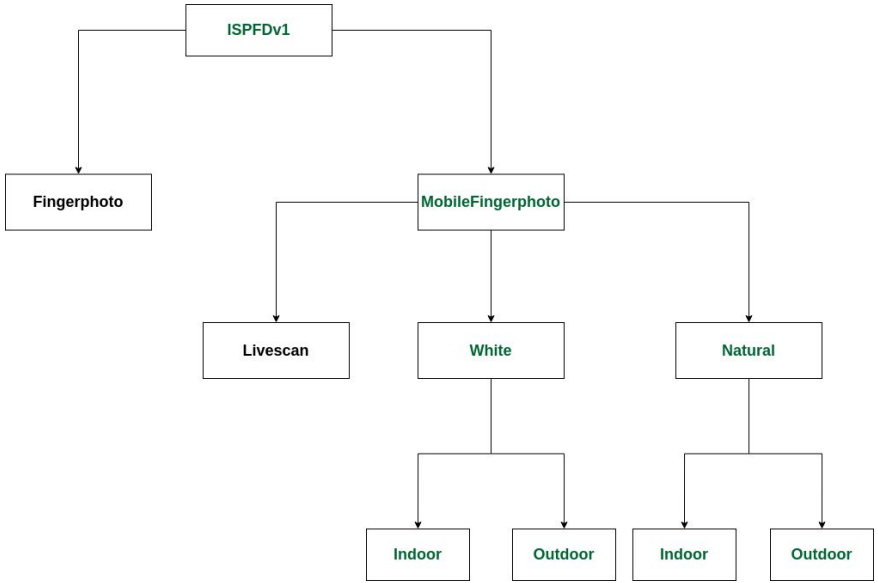
- **Contactless fingerprint recognition** is the latest step in the evolution of fingerprint recognition.
- There is no need to press a hand against a capturing device.
- It's only required:
  - A hand;
  - A camera;
  - A special app to record and analyze fingerprint.
- Our goal in this project was to **construct** a contactless fingerprint-based biometric system and to **analyse** its effectiveness with various feature extraction algorithms.

Sources of complexity

- The contactless approach is way harder than the traditional approaches due to:
  - quality of input image;
  - variability of pose and orientation;
  - noise or out of focus;
  - illumination conditions;
  - segmentation from the background;
  - ROI extraction;
  - conversion to a livescan-like image;
  - no established OTS applications;
  - …

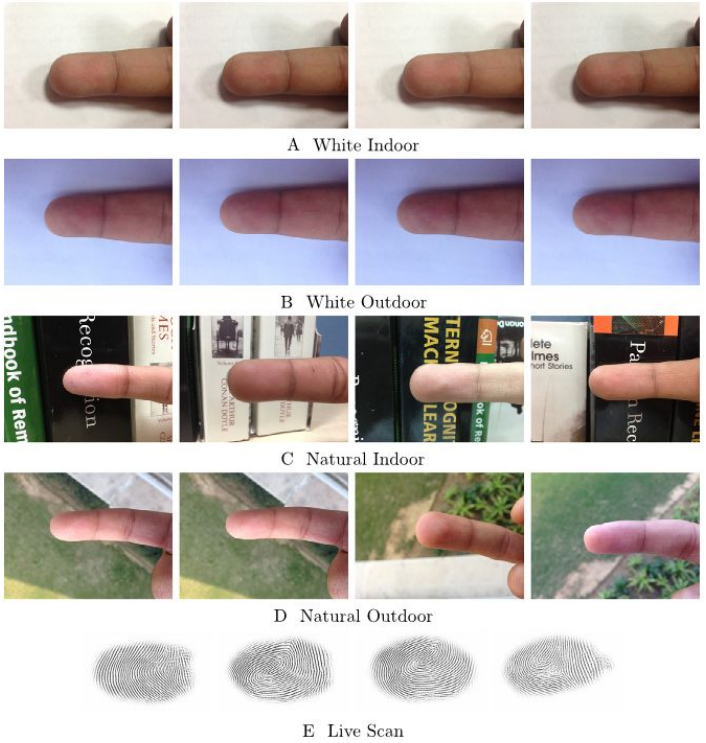
Dataset

- For the development of the project we used the IIITD SmartPhone Fingerphoto Database v1 (**ISPFDv1**).



Dataset (cont'd)

- In each of the 4 subfolders there are 64 subjects and for each of them there are 16 photos:
  - 8 for the right middle finger;
  - 8 for the right index finger.
- Each photo presents the same name pattern: five elements divided by an underscore (e.g., "3_i_1_w_3.jpg"):
  - The first element represents the subject ID;
  - The second element indicates whether the photo was taken indoor (i) or outdoor (o);
  - The third element indicates which finger is present in the photo;
  - The fourth element indicates the background (white or natural);
  - Last element is the sample number.

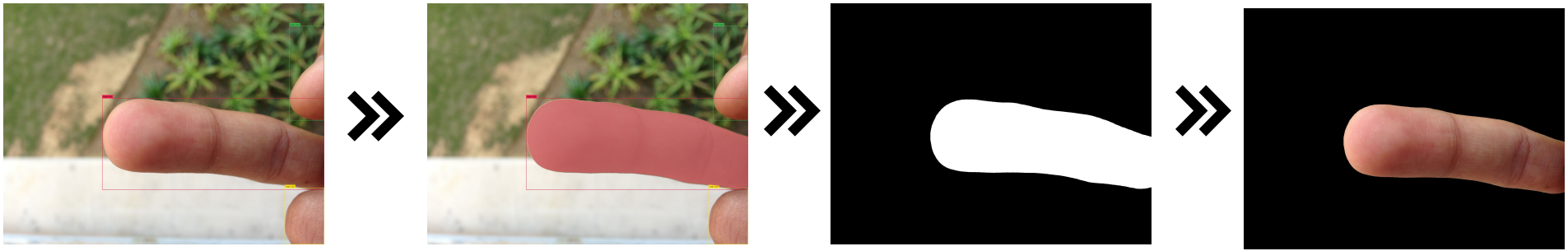


Preprocessing pipeline

- Before the fingerprint recognition process can take place, there are different preprocessing steps that are needed in order to get a suitable representation of the fingerprints:
  - finger detection and segmentation;
  - fingertip cropping;
  - fingerprint enhancement;
  - dataset division.

Detection and Segmentation

- **Object Detection** is a computer vision task that involves identifying and localizing objects of interest within an image or a video. The goal is to detect and classify objects of various classes, while also providing their precise bounding box coordinates;
- **Semantic Segmentation** is the task of classifying each pixel in an image into predefined semantic categories or classes;
- A successful approach was developed using **Grounded-SAM**, the combination of **Grounding Dino** and **SAM (Segment Anything Model).**



Detection and Segmentation (cont'd)

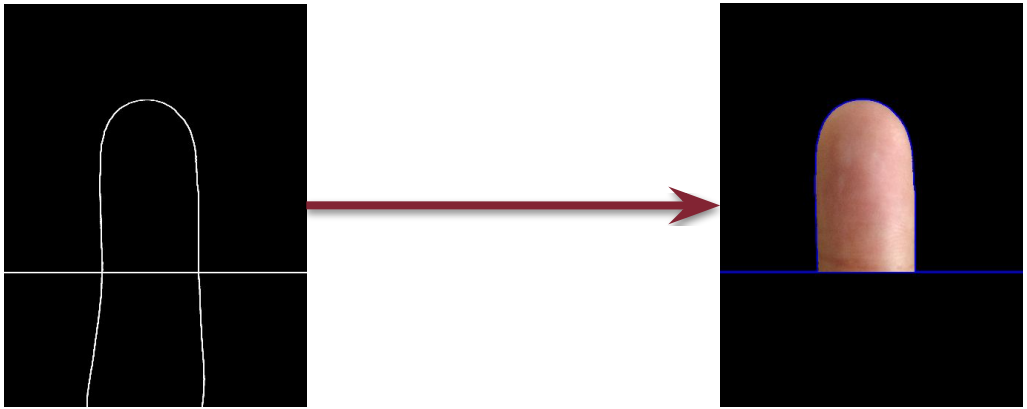
- **Grounding Dino**: a pretrained transformer model for detecting objects in images by conditioning on various input modalities, in our case we conditioned the detection via the text prompt **"finger"**;

- **SAM (Segment Anything Model)**: a pretrained transformer model from META, capable of segmenting any object in an image. Since it can be conditioned with a bounding box we conditioned the segmentation via the bounding box returned by Grounding Dino, in order to obtain the segmentation mask only for the main finger in the image.

Fingertip cropping

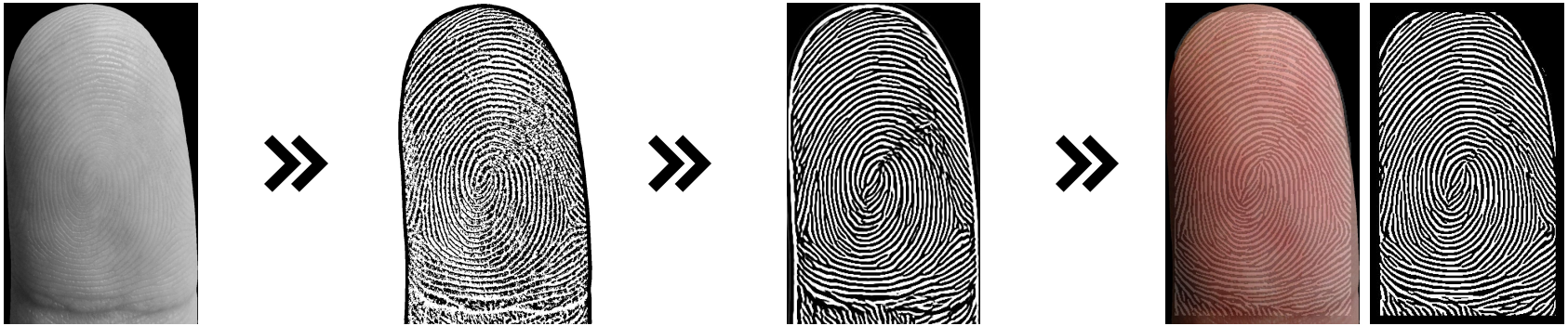
- The objective of this phase is to extract from the segmented images only the **ROI**, which in our case is the **fingertip** (i.e. the area that approximately goes from the tip of the finger to the end of the first phalanx);

- Our approach follows several steps: image rotation and grayscale conversion, contrast enhancement, thresholding, canny edge detection, dilation and erosion, search and cut out of a minimum area rectangle with predefined height (heuristic) that encompasses the edged area above the horizontal line.



Fingerprint enhancement

- The goal of this part is to **convert** the fingertip image into a livescan-like image, so that we can proceed with the fingerprint matching part;
- Also in this case we performed several computer vision steps, including:
  - Grayscale conversion;
  - gamma correction;
  - adaptive mean thresholding;
  - image resize and Gabor filtering;
  - "Fingerprint-Enhancement" library application.



Quality assessment

- After the enhancement, we wanted to check the quality of our obtained fingerprints;
- This was done by using the **NIST Fingerprint Image Quality (NFIQ) 2;**
- For each dataset subfolder we computed the average quality scores:

Folder	Avg. quality score
Natural/Indoor	44.3
Natural/Outdoor	**48.9**
White/Indoor	41.8
White/Outdoor	**47.6**

Dataset division

- Originally we took into account all the 4 subfolders, but this led us to impractical times for the matching phase (i.e. BOZORTH3 ~20 hours). So we kept only the outdoor subfolders (higher average quality) and only the index for each subject;

- **Train** (70%) vs **test** (30%) split, useful to find the best threshold over the training set and then test that threshold one final time on the held-out test set. Simulates the case in which a system is put into production;

- **Probe** vs **gallery** division both over the train and test set:
  - train probe set: 5 samples per 64 identities, with a total of 320 samples;
  - train gallery set: 6 samples per 64 identities, with a total of 384 samples;
  - test probe set: 2 samples per 64 identities, with a total of 128 samples;
  - test gallery set: 3 samples per 64 identities, with a total of 192 samples.

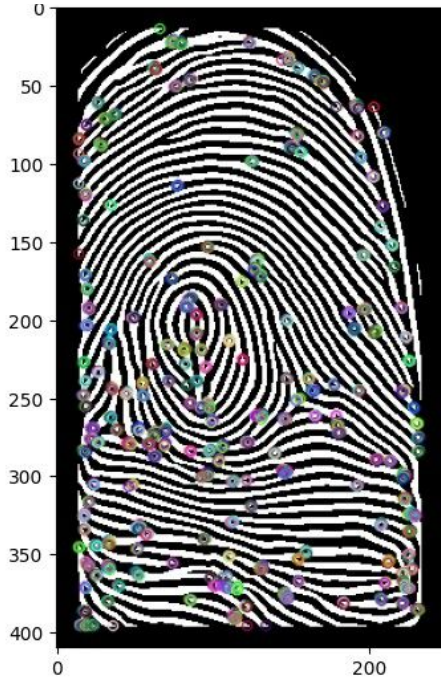
- Probe vs gallery partitioning for the test set was repeated at random 5 times and the final metrics that we will compute are the averages over these 5 splits.

Fingerprint recognition

- Once completed all the preprocessing steps, we were able to focus on the recognition task.
- In particular, we tried three different methods that are respectively based on the following feature extraction algorithms:
  - SIFT;
  - ORB;
  - MINDTCT.
- For all these approaches we simulated the **identification open set** operation by using the **all probe vs all gallery** evaluation matrix.
- The **metrics** we intended to calculate are those typical of the open set identification task:
  - Detection and Identification Rate at Rank 1;
  - Genuine Rejection Rate;
  - False Acceptance Rate;
  - False Rejection Rate;
  - Equal Error Rate;
  - ROC curve and AUROC.

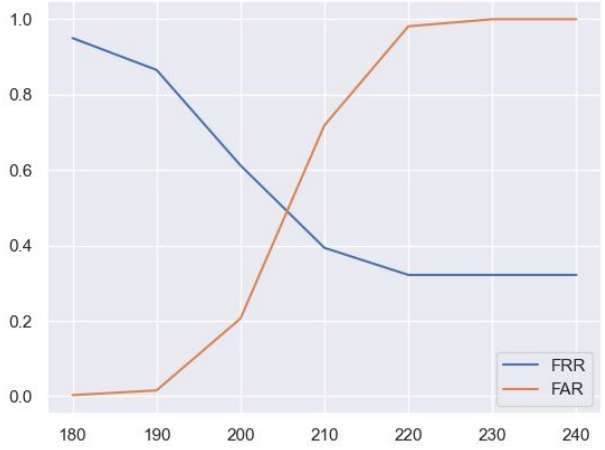
SIFT method

- **SIFT** is an approach for extracting distinctive features from images which are invariant to rotation and scale;
- It's a technique for detecting salient, stable feature points in an image and for every such point it also provides a set of features that characterize a small image region around it;
- Once we computed the keypoints descriptors, we used the OpenCV **BF-Matcher** (Brute-Force Matcher): a simple matching algorithm that compares each feature descriptor in one set of features with all the feature descriptors in another set of features and finds the top-k best matches based on a distance metric.

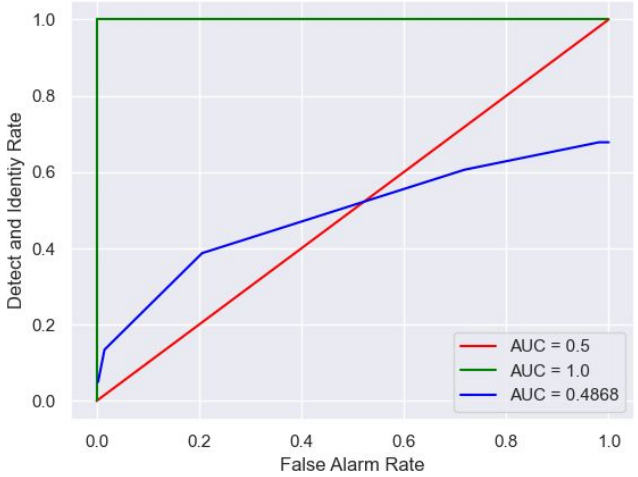


SIFT evaluation

FAR vs FRR curves



Watchlist ROC and AUROC



Test results

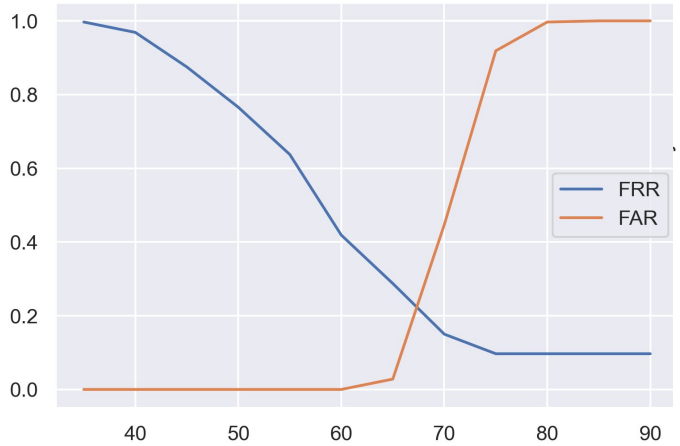
Evaluation metrics	Results
DIR(1)	0.351 ± 0.005
FAR	0.434 ± 0.027
GRR	0.565 ± 0.027
FRR	0.648 ± 0.005

ORB method

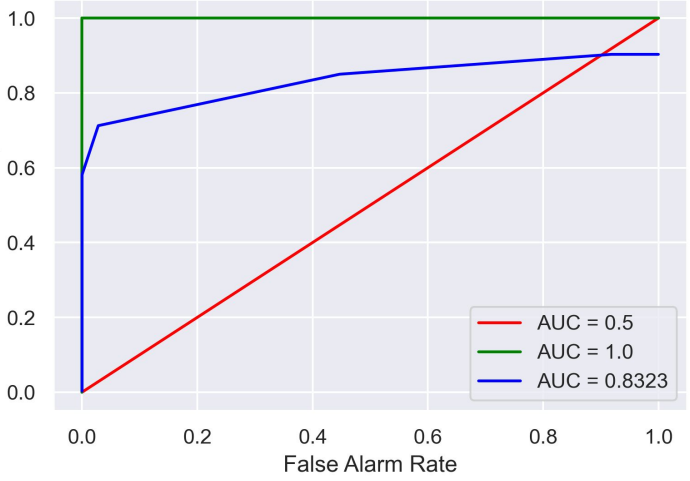
- **ORB** is a good alternative to SIFT in terms of computation cost and matching performance;
- It is basically a fusion of **FAST keypoints detector** and **BRIEF descriptors** with many modifications to enhance the performance;
- First, it uses FAST to find keypoints, then applies Harris corner measure to find top N points among them and finally it uses pyramid (a technique that consists of repeatedly downsampling the original image to produce a series of reduced-resolution versions) to get multiscale-features;
- As in the SIFT approach, to compare the descriptors of two images we used a BF-matcher.

ORB evaluation

FAR vs FRR curves



Watchlist ROC and AUROC



Test results

Evaluation metrics	Results
DIR(1)	0.762 ± 0.008
FAR	0.235 ± 0.009
GRR	0.764 ± 0.009
FRR	0.237 ± 0.008

MINDTCT-BOZORTH3 method

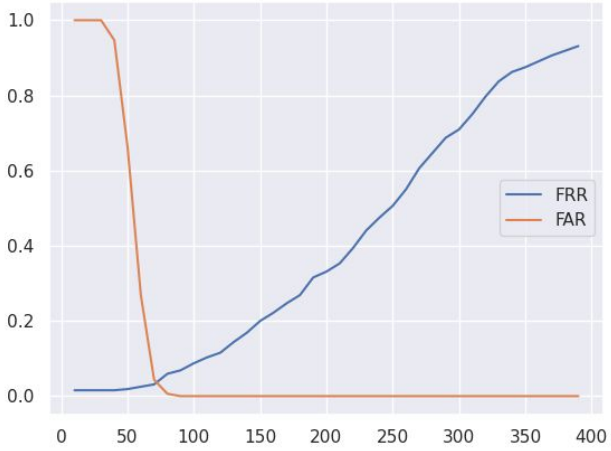
- The third approach we used comes from the NIST Biometric Image Software **(NBIS)**;
- It proposes **MINDTCT** as the minutiae extractor and **BOZORTH3** as the feature matcher;
- The minutiae are first extracted from the two fingerprints through MINDTCT, that allowed us to obtain the orientation and the location of the minutiae that we stored as two sets of points in a two dimensional space;
- After that, BOZORTH3 searches for the alignment between the sets that maximizes the number of corresponding pairs of minutiae for the fingerprint matching system.

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

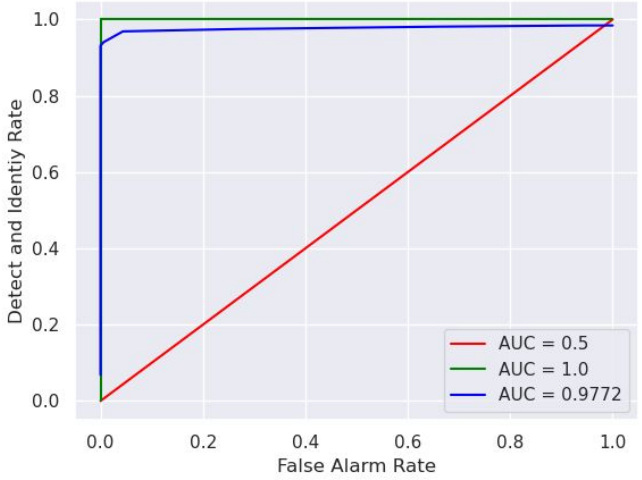
Link here!

BOZORTH3 evaluation

FAR vs FRR curves



Watchlist ROC and AUROC



Test results

Evaluation metrics	Results
DIR(1)	0.897 ± 0.011
FAR	0.017 ± 0.006
GRR	0.983 ± 0.006
FRR	0.103 ± 0.011

We made it look easy but…

- Many other things were tried and discarded because they didn't achieve the desired results:
  - U-Net for detection and segmentation of the finger;
  - CNN for fingertip cropping;
  - sharpening, histogram equalization, image enhancement and coherence filtering for fingerprint enhancement;
  - Seam Carving for resizing without deforming;
  - Minutia Cylinder-Code and Fingerflow for features extraction.

Conclusions and future works

- Thanks to this project, we were able to get into a very modern task with which we learned and used very powerful techniques;
- Although it was a very complicated task, we managed to achieve very good performance on a subset of the ISPFDv1 dataset;
- We found that matching based on minutiae gives us better performance than methods based on extracting local keypoints;
- As a possible future work it could be scaled on the entire dataset and the entire process that we have implemented could be integrated in a real system.

SAPIENZA
UNIVERSITÀ DI ROMA

Demo

Link here!

Fingerprint Matching Demo

This demo aims to compare the performance of three different fingerprint matching algorithms: MINDTCT, SIFT, and ORB.

To participate in the demo, follow these steps:

1. **Run the provided code**: Execute the code to initialize the fingerprint matching environment.
2. **Choose a fingerprint**: Select either a fingerprint from the probe or the gallery.
3. **Select a matching method**: Decide which algorithm to use for the matching process. You can choose between MINDTCT, SIFT, and ORB.
4. **Click the match button**: After selecting the fingerprint and the matching method, click the "Match" button to initiate the comparison.

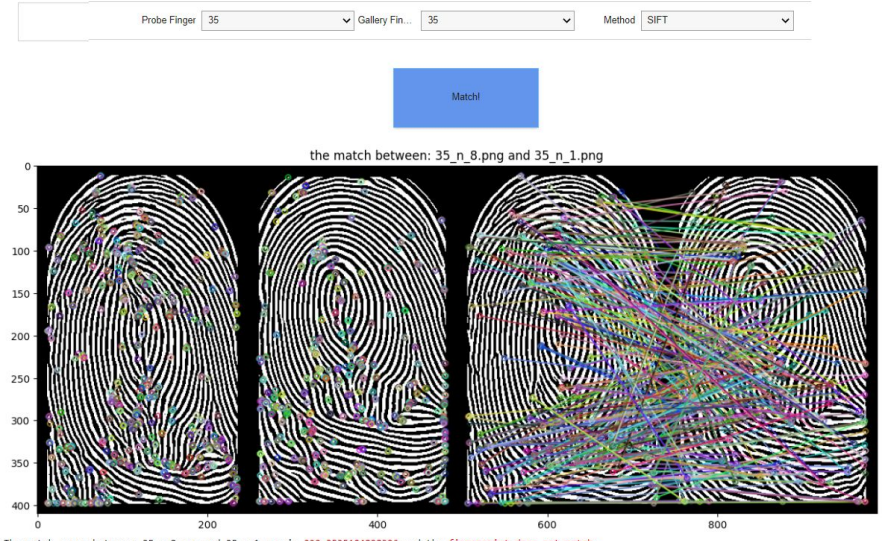
Stay tuned for the results and insights gained from this exciting fingerprint matching demo!

Let the comparisons begin!

0. Preliminaries

Import Libraries

```
[9] from skimage import io
    import os
    import cv2
    import sys
    import numpy as np
    import json
    import matplotlib.pyplot as plt
    import ipywidgets as widgets
    from IPython.display import display
    import subprocess
    from skimage import io
```



Thank you for the attention!