Contactless fingerprint recognition
Biometric Systems course a.y. 22/23
Project report

vannoli.1837649@studenti.uniroma1.it[1], santoro.1843664@studenti.uniroma1.it[1], palma.1837493@studenti.uniroma1.it[1], and manganaro.2017897@studenti.uniroma1.it[1]

[1]Sapienza University of Rome

June 19, 2023

Contents

1 Introduction

In today's digital age, security has become a paramount concern across various industries and sectors. One of the most reliable and widely used methods of personal identification and authentication is through biometric systems. Among the different biometric modalities, fingerprint recognition stands out as a highly accurate and efficient technology. Fingerprint biometric systems have gained significant popularity due to their proven reliability, ease of use, and robustness against fraudulent attempts.
A fingerprint biometric system is a technological marvel that captures, analyzes, and compares unique patterns present on an individual's fingertips to establish their identity. The human fingerprint has been recognized as an inherently distinct and exclusive feature for each individual, making it an ideal biometric characteristic for identification purposes because the pattern of ridges and furrows on the fingertip provides an intricate map that is virtually impossible to replicate accurately. Minutiae points are, instead, the locations where a ridge becomes discontinuous. A ridge can either come to an end, which is called as termination or it can split into two ridges, which is called as bifurcation. These local keypoints are different in each person and are very discriminative.
Contactless fingerprinting is the latest step in the evolution of biometrics. Unlike the inkpad and bioscanner methods, the setup for contactless fingerprinting removed the need to press a hand against a device. This technology only requires a hand, a camera, and a special app to record and analyze fingerprints. The contactless approach is way harder than the traditional livescan-based approaches because the performances may be affected by a lot of factors (such as the quality of the image, the variability in terms of pose and orientation, the noise, introduced for example by illumination conditions, and so on) and because it requires a lot of preprocessing in order to reconstruct the same fingerprint that we would obtain with a livescan. In this project, by means of a dataset of finger images, we have, through various steps that will be explained below, constructed a contactless fingerprint-based biometric system and analysed its effectiveness with various feature extraction algorithms.

Figure 1: On the left a touch-based fingerprint (also called livescan), on the right a touchless fingerprint image

2 Dataset

For the development of the project we used the IIITD SmartPhone Fingerphoto Database v1 (ISPFDv1) [15], which was kindly granted to us by Professor De Marsico.
It is composed by two main folders:

- **Fingerphoto:** images of one or two fingers taken with a camera in a completely unconstrained setup;
- **MobileFingerphoto:** single finger images in horizontal position.

Figure 2: An example of a photo taken from the dataset MobileFingerphoto

Figure 3: An example of a photo taken from the dataset Fingerphoto

We decided to use the **MobileFingerphoto** part, because it is less wild and more suited for our task. **MobileFingerphoto** is composed by three folders:

- **White:** images of fingers with a white background.
- **Natural:** finger images with variable background.
- **Livescan:** livescan fingerprint images.

The last folder was discarded while we used the first two. The reason of this choice lies in the fact that we want to obtain images as similar as possible to live scans starting from images captured via a mobile phone in different conditions, with the aim of recognizing the fingerprint's identity. Both **Natural** and **White** consist of two subfolders called **indoor** and **outdoor**: in the former setup images were taken in an indoor environment while in the latter those were taken in an outdoor environment.
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 and 8 for the right index finger, so we have a total of 4096 images. 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 (1 stands for right index finger, 2 stands for right middle finger);
- The fourth element indicates the background (w is white and n is natural);

- The last element indicates the sample number for that subject.

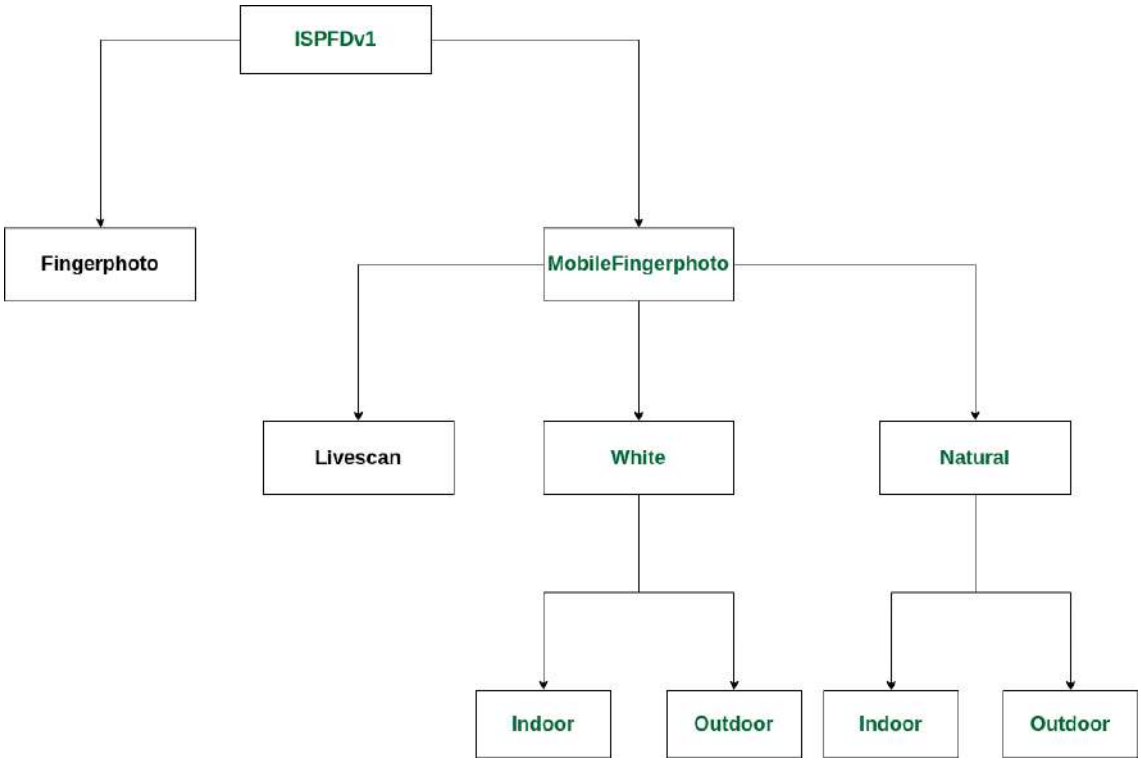

Figure 4: The organization of the ISPFDv1 dataset, in green what we selected for our project.

3 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, including:

- finger detection and segmentation;
- fingertip cropping;
- fingerprint enhancement;
- dataset division.

3.1 Detection and segmentation

Object **Detection** is a computer vision task that involves identifying and localizing multiple 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. The goal is to assign a single label to every pixel in the image, representing the category to which it belongs. This means that all pixels belonging to the same object or region in the image will have the same label. Semantic segmentation focuses on understanding the overall structure and content of the image by labelling different regions based on their semantic meaning. In our application we need to detect and segment the main finger from the photos because we don't care about all the other informations that are present in the background.

At first we tried to use the U^2Net network [12] to detect and segment in one step. It is a deep architecture with a two-level nested U-structure designed for Salient Object Detection, which is the task of detecting the most visually distinct and attention-grabbing objects in an image. As we can see in image 5, this approach didn't give us the desired results as some fingers were poorly segmented.

Figure 5: Example of a poor finger segmentation with U^2Net.

A successful approach was developed using **Grounded-SAM**, which is the combination of:

- **Grounding Dino[9]:** a pretrained transformer model for detecting an object by conditioning on various input modalities, in our case we conditioned the detection via the text prompt "finger";
- **SAM (Segment Anything Model)[7]:** a pretrained transformer model 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.

Hence **Grounded-SAM** allows us to detect and segment the finger from a textual input. Since Grounding Dino can return more than one bounding box, we just keep the one with the highest probability score. Finally, we apply the retrieved binary mask to the original image in order to keep only the finger and eliminate the background.

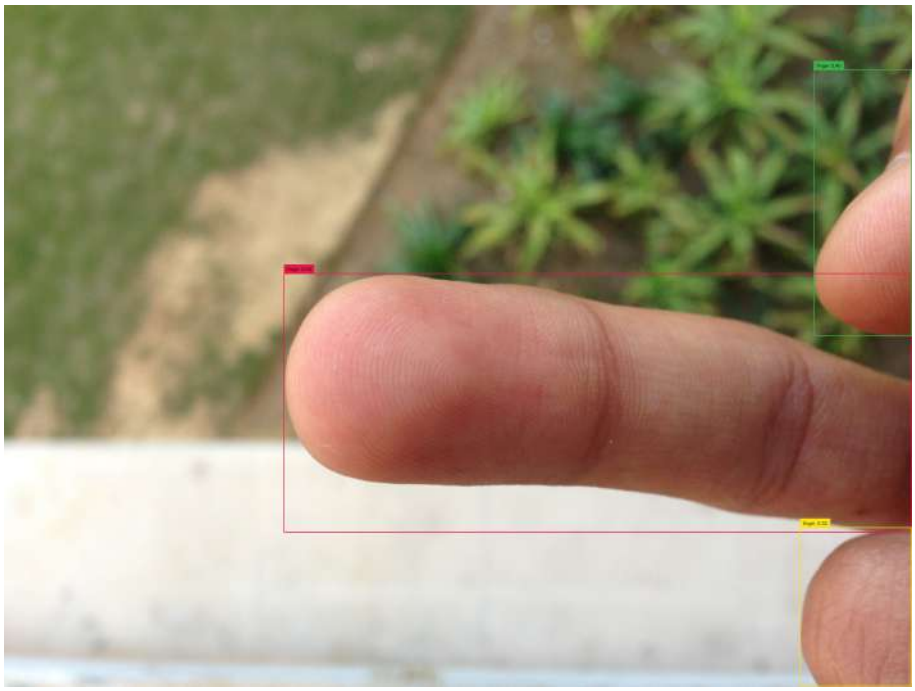

Figure 6: Finger detection with Grounding Dino and text prompt "finger".

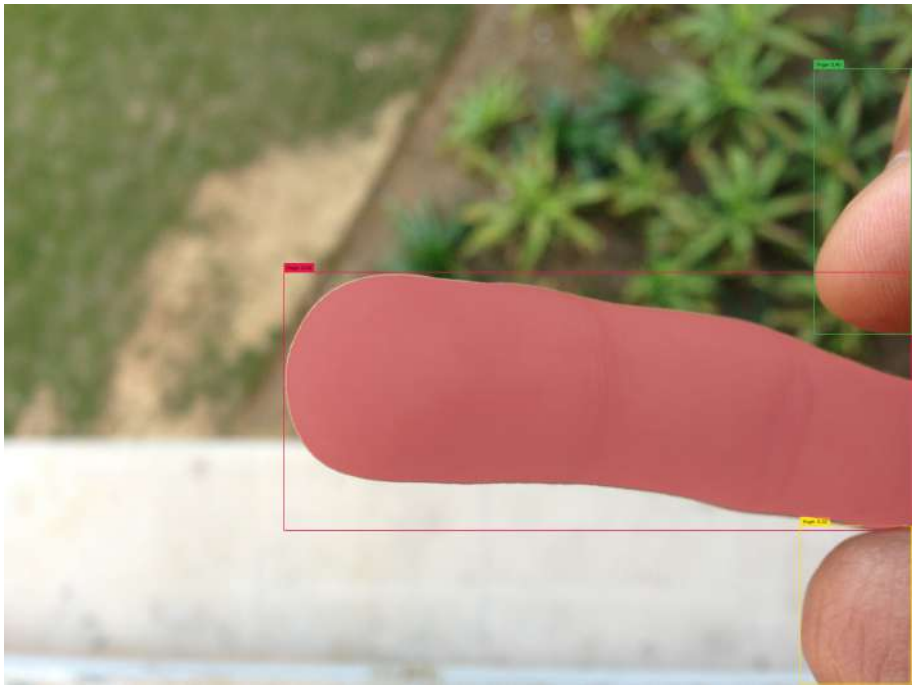

Figure 7: Finger segmentation with SAM.

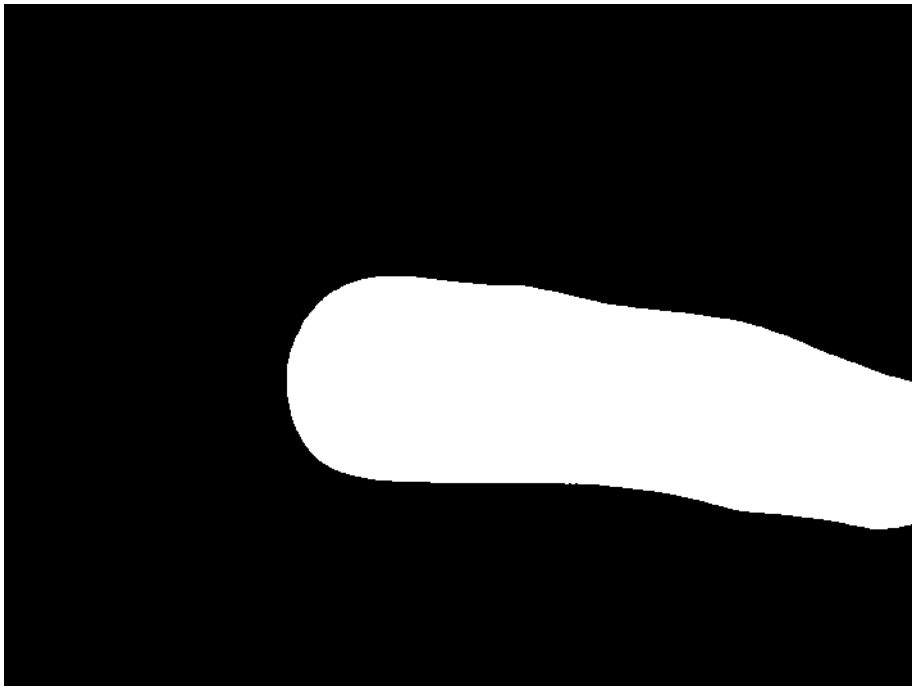

Figure 8: Mask application for background removal.

Figure 9: Segmented finger.

3.2 Fingertip cropping

The objective of this phase is to extract from the segmented images only the ROI (Region Of Interest), 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). To obtain it there were mainly two solutions, either to use deep learning or to opt for computer vision.
At first we tried to use the **PAD CNN**, a segmentation algorithm based on a CNN method trained specifically on the ISPFDv1 dataset. The authors claim that it should be able to detect the fingertip area from colored images and provide a cropped region of interest for further processing for the contactless fingerprint recognition task. As we can see in image 10, this approach didn't give us the desired results as some fingertips were poorly extracted.

Figure 10: Example of a poor fingertip extracted with PAD CNN, too large an area was taken under the first phalanx and too many black pixels are on the left side.

The winning approach was obtained through computer vision techniques and was developed starting from the **Fingertip-Detection-With-OpenCV** github repository. There are several steps to be followed to obtain the fingertip:

- initially the segmented image is rotated by 90° to the right and we convert it to grayscale because color information does not help us find the edges of the fingertip;
- increase its contrast to better differentiate the foreground from the background and blur it to keep only the biggest edges;
- threshold to select only the brightest pixels of the image, which should be only the finger since we increased the contrast between the brightest and dimmest pixels;
- apply the Canny edge detection algorithm;
- dilate first vertical then horizontal lines, in order to connect possible gaps in the hand outline. Then erode in a cross pattern to remove smaller lines and inconsistencies, leaving only the outline of the finger;
- once we have the outline we cut to a fixed height of the image as shown in Figure 11, then we move the line up or down until the minimum area rectangle that encompasses the edged area above the line has height < 1150 pixels (heuristic measure);

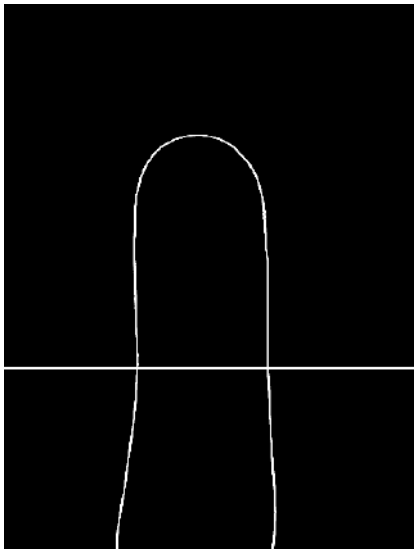

Figure 11: We cut the eroded image up to a predefined height (i.e. the horizontal bar) and start iterating until we find the fingertip.

- after having found the right height, we cut the minimum area rectangle that encompasses the edged area above the line. This will be our fingertip.

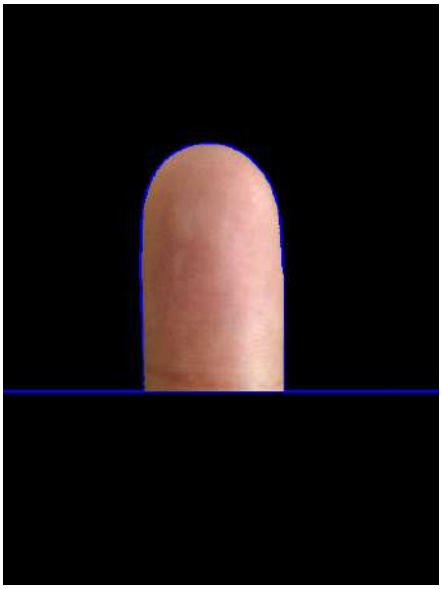

Figure 12: Final ROI detected by our algorithm.

Figure 13: Same fingertip as in Figure 10, but extracted with our method.

3.3 Fingerprint enhancement

This part of the pipeline has the goal of converting the fingertip image into a livescan-like image, so that we can proceed with the fingerprint matching part. There are several steps to be followed to obtain the fingerprint:

- firstly we convert the image to grayscale, because again color information does not help us find the edges of the fingertip;
- apply gamma correction, a technique used in computer vision and image processing to compensate for the non-linear relationship between the pixel values of an image and the perceived brightness

by the human visual system. To perform gamma correction, each pixel value in an image is raised to the power of the reciprocal of the γ value. Mathematically, the function is as follows:

$$CorrectedValue = OriginalValue^{\frac{1}{\gamma}} \tag{1}$$

with $\gamma = 1.42$ in our case. In this way we mitigate shadows and improve the visibility of details in darker areas;

Figure 14: Gamma correction applied.

- as reported in [11], Adaptive Mean Thresholding (AMT) is one of the most common techniques used to extract the pattern of ridges and furrows, so we applied it. It is a technique used in image processing and computer vision to segment an image into binary regions based on local pixel intensities. Unlike global thresholding, which applies a single threshold value to the entire image, Adaptive Mean Thresholding calculates a local threshold for each pixel based on the average intensity of its neighborhood;

Figure 15: AMT applied.

- resize all images to same dimension (average dimension / 3) and, since AMT extracts non-continuous lines, apply an **enhancement algorithm** written by the **Biometric System Laboratory** of University of Bologna in order to improve the quality of the fingerprint. First they find the local ridge orientation according to [3] and the local ridge frequency according to [6]. Then, in order to enhance the fingerprint pattern, a convolution with a bank of 8 Gabor filters with different angles is performed, all of them having the same period. The final image is assembled taking the right pixel from each convolved image, using the discretized orientation in the input image as a guidance;

Figure 16: Gabor filtering applied.

- since some lines that should be contiguous in the fingerprint extracted are still not contiguous, we apply the **Fingerprint-Enhancement-Python** library (which is again based on oriented Gabor filters) and obtain our final fingerprint.

Figure 17: Final fingerprint image.

Figure 18: Overlapping the segmented fingertip with the extracted fingerprint we can see that the extracted patterns are reliable.

We have to mention that a lot of other computer vision algorithms were tried and then discarded, because they did not improve the results or even made them worse. Some of those are: convolution with sharpening kernel, unsharp masking, histogram equalization, CLAHE, DHE [1], Ying enhancement [17], coherence filtering, Seam Carving [2].
Finally, a quality assessment was carried out in order to check the quality of our obtained fingerprints. This was done by using the **NIST Fingerprint Image Quality (NFIQ) 2** [16] tool from NIST, which produces quality scores in the range [0, 100]. The average quality scores are as follows, please notice that the quality of many images in the ISPFDv1 dataset is very poor (mainly due to out of focus pictures), and none of the preprocessing steps we applied degraded the quality of the original image:

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

Table 1: Average quality scores computed by NFIQ2.

3.4 Dataset division

According to the standard datasets division, we decided to divide our dataset into a train set and a test set. Originally we took into account all the 4 subfolders reported in Table 1, but this led us to impractical times to perform the matching of the fingerprints (i.e. BOZORTH3 required ∼20 hours). So we decided to keep only the **outdoor** subfolders because they have higher average quality, after unifying those two we only kept the right index for each subject. In this way we have a total of 16 samples per 64 identities, with a total of 1024 different samples.
Even if we will not use feature extraction methods that require an actual training, the train vs test division is essential in order to find the right threshold on the train set and then test the performance of the found threshold on the test set. This is also useful in order to: simulate the case in which a system is put into production with a predefined threshold; have a fair comparison of different fingerprint matching algorithms, as the test set is "new and never seen" for everyone.
70% of the data (i.e. 11 samples per identity) goes into the train set and 30% of the data (i.e. 5 samples per identity) goes into the test set, then both of these sets were divided into probe and gallery. The final dataset is organized as follows:

- **train set**: a total of 704 samples divided into:
  - **probe set**: 5 samples per 64 identities, with a total of 320 samples;
  - **gallery set**: 6 samples per 64 identities, with a total of 384 samples;
- **test set**: a total of 320 samples divided into:
  - **probe set**: 2 samples per 64 identities, with a total of 128 samples;
  - **gallery set**: 3 samples per 64 identities, with a total of 192 samples.

The probe vs gallery division 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, in order to ensure reliability of the metrics. The evaluation approach that we used is the **all probe vs all gallery**, as it is described in the course slides, the task we implemented is **identification open set**, because it is the most challenging task for a biometric system.

4 Fingerprint recognition

Once completed all the preprocessing steps, we were able to focus on the use of methods that allow us to perform the fingerprint recognition. We tried different approaches that consist first in the extraction of the features from each image, and then in the measurement of the distances (or similarities) between each pair of probe-gallery image. 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 followed the same testing procedure: identification open set using the all probe vs all gallery evaluation matrix. First, we used this matrix over the training set in order to find the best threshold value, which we used later over the 5 test set spits to find the final results. 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.

Since false acceptances can only occur if we consider impostor attempts, for each probe we decided to simulate, in turn, a genuine attempt and an impostor attempt.

4.1 SIFT method

The first approach we followed to extract the features from the fingerprint images is SIFT [10]. The original SIFT algorithm proposed by Lowe is an approach for extracting distinctive features from images which are invariant to rotation and scale. So it is 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.

Figure 19: Example of the keypoints detected on a fingerprint image with SIFT.

Starting from the features relative to the keypoints of a probe image, we had to measure the distance with the features of each gallery image. To achieve this goal, we used the **OpenCV** BF-Matcher (Brute-Force Matcher): it is a simple and straightforward 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. In this case, as a distance metric we used an Euclidean distance (the L2 distance).

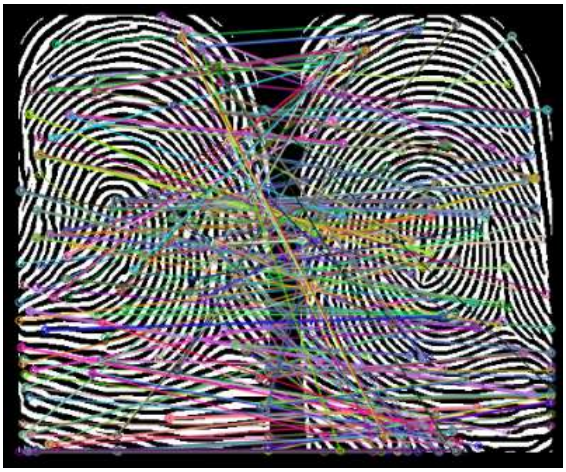

Figure 20: Example of matching between two set of fingerprints keypoints using SIFT.

So, we matched all probe templates with all gallery templates, we computed the metrics that we mentioned previously on the train set and we plotted the results.

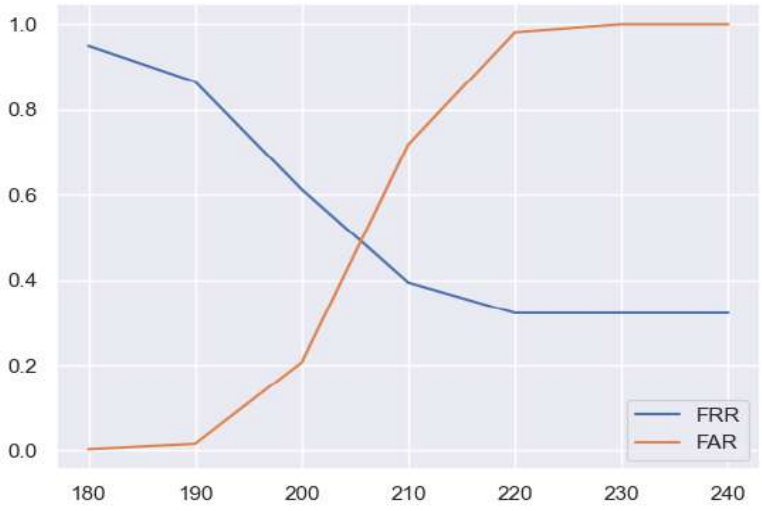

Figure 21: FAR vs FRR curves obtained using SIFT.

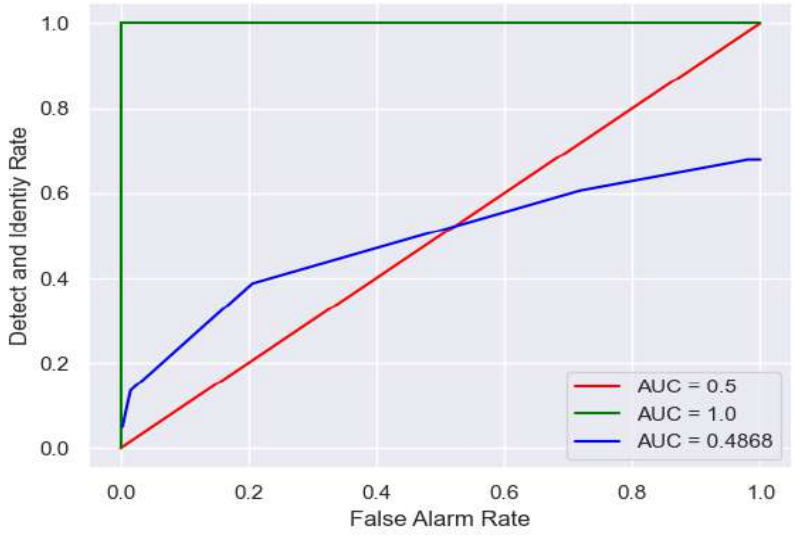

Figure 22: Watchlist ROC and AUROC using SIFT.

Through these plots we have noticed that the SIFT-based approach performs very poorly, with an Equal Error Rate of around 0.5 and an AUROC of around 0.48. One of the reasons why SIFT does not perform very well may lie in the fact that it is primarily designed for object recognition in images with distinctive visual features, such as sharp corners, edges and blobs. However, fingerprint images have unique patterns characterized by small ridges, valleys, and minutiae points, which are not the same type of features that SIFT is designed to detect. The plot of the FAR vs FRR allowed us to choose the threshold value which is 205 and corresponds to the EER. We repeated the same tests over the 5 splits of the test set by using this fixed threshold value and obtained the following 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

Table 2: Test results obtained with SIFT.

4.2 ORB method

Another approach that we used to get the descriptors of the fingerprints is based on the ORB algorithm (Oriented Fast and Rotated BRIEF) [14]. As described in the paper, it is a good alternative to SIFT in terms of computation cost and matching performance. ORB is basically a fusion of FAST keypoints detector [13] and BRIEF descriptors [4] 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. In this case we based our code on an existing github repository, from which we just took part of the code inherent to feature extraction and matching, to reach the goal we were interested in. As in the SIFT approach, to compare the descriptors of two images we used a BF-matcher that in this case is based on the Hamming distance rather than an Euclidean one.

Figure 23: Example of matching between two sets of fingerprints keypoints detected using ORB.

At this point, we were able to fill up the distance matrix and then we focused on choosing the thresholds to be used to calculate the evaluation metrics on the train set. Therefore, to do that, we plotted the distribution of the scores obtained by comparing the probe set and the gallery set.

In the following plots we can observe that the scores are concentrated roughly in the range from 35 to 85, so we have selected a number of thresholds that cover the whole distribution. At this point we performed the identification open set task and plotted the results obtained over the training set.

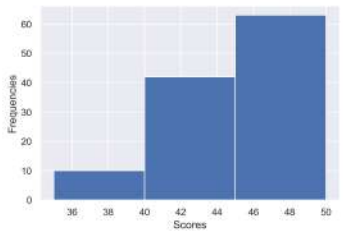

Figure 24: Scores distribution in the range 35-50.

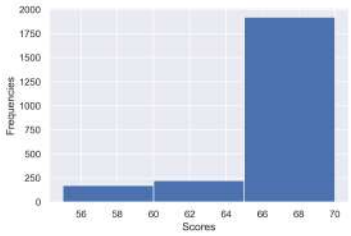

Figure 25: Scores distribution in the range 50-75.

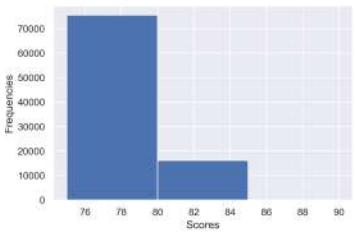

Figure 26: Scores distribution in the range 75-90.

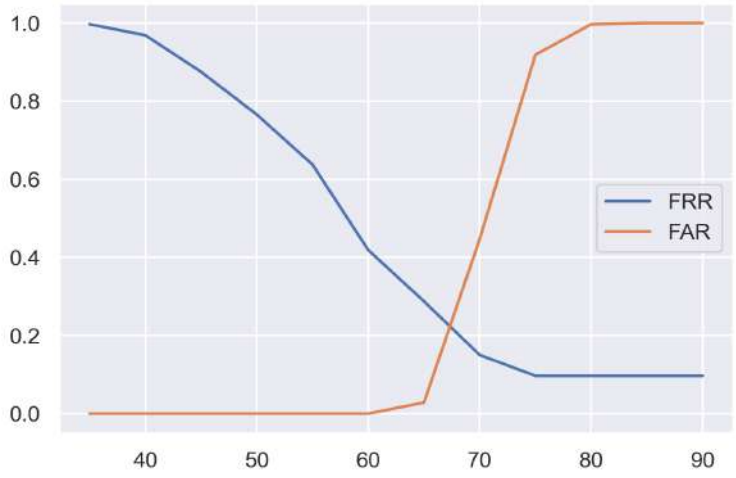

Figure 27: FAR vs FRR curves using ORB.

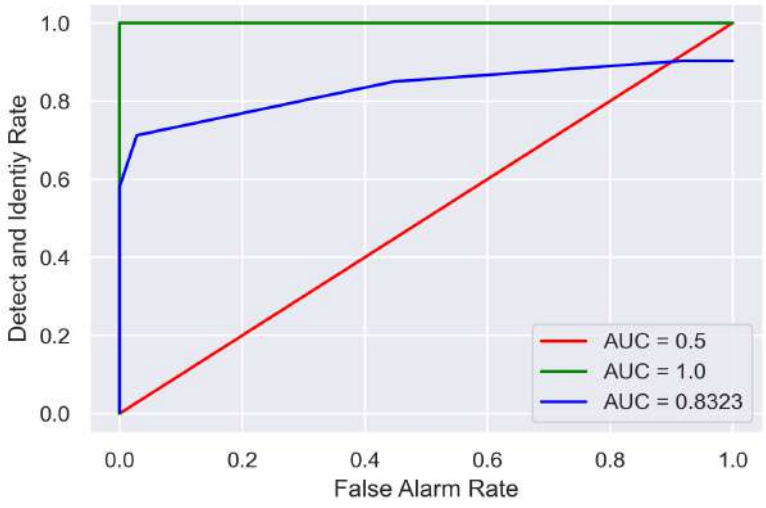

Figure 28: Watchlist ROC and AUROC using ORB.

Looking at the first plot we can see that as we increase the threshold value, the False Acceptance Rate and the False Rejection Rate increases and decreases respectively in a good way. We can also see that we obtained a good Equal Error Rate value (around 0.2) and a good Area Under ROC Curve value (around 0.83). The plot of the FAR vs FRR also allowed us to choose the best threshold value, which is 69 and corresponds to the EER. We repeated the same tests over the 5 splits of the test set by using this fixed threshold value and we obtained the following 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

Table 3: Final results obtained with ORB.

4.3 BOZORTH3 method

The third approach we used comes from the NIST biometric image software (NBIS) [8]. In detail, NBIS 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.
Unlike the approaches seen before, BOZORTH3 uses a minutiae-based matching approach (taking into account orientations and distances of minutiae of two different fingerprints) and returns a similarity score, not a distance one.
As for the previous approaches, we plotted the False Acceptance Rate curve versus the False Rejection Rate curve, and the ROC curve on the train set.

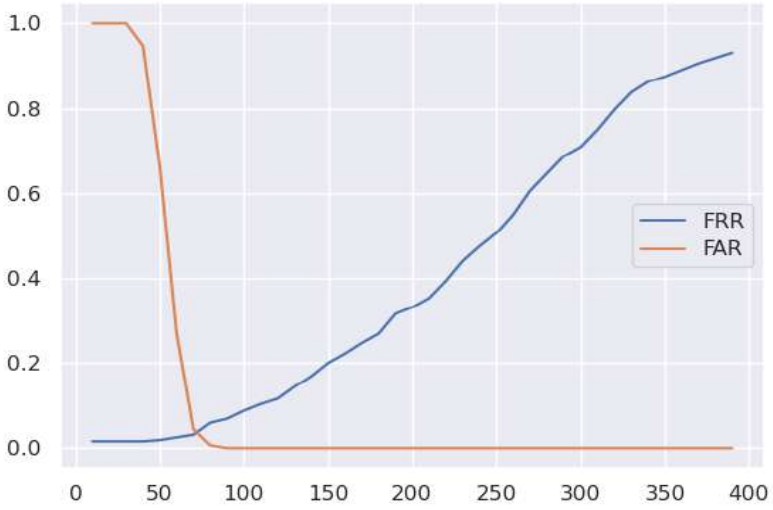

Figure 29: FAR vs FRR curves using BOZORTH3.

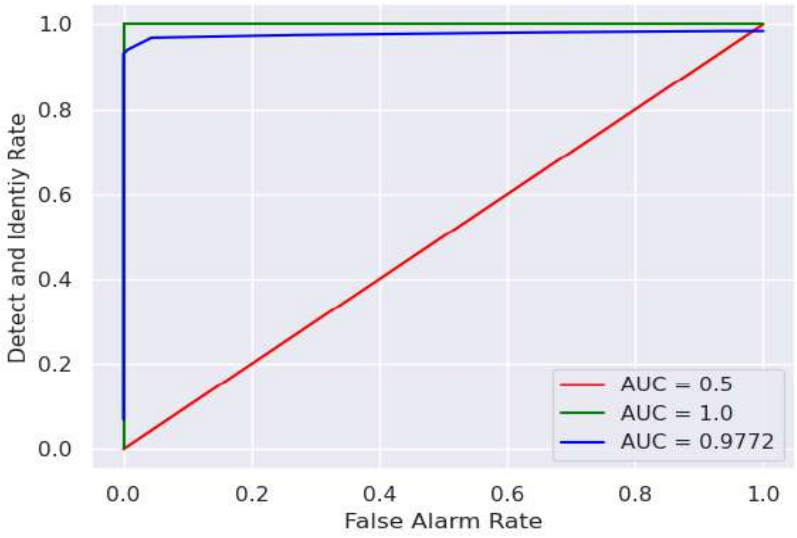

Figure 30: Watchlist ROC and AUROC using BOZORTH3.

As we can see from the plots, MINDTCT and BOZORTH3 is a particularly well performing combo, in fact we obtained a very low Equal Error Rate (around 0.04 at threshold value 72) and a very high Area Under ROC Curve (around 0.97). The reason for this high results could derive from the fact that it is very robust against translations and rotations. In this case it is important to point out that the FRR value increases if we increase the threshold and the FAR value decreases if we increase the threshold, this is due to the fact that in this case we do not have anymore distance scores, but similarity scores. We finally repeated the metrics computation over the test set, the results are shown below:

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

Table 4: Test results obtained with BOZORTH3.

It is worth mentioning that we also tried other feature extraction approaches, such as: Minutia Cylinder-Code [5], the library Fingerprint-Feature-Extraction and Fingerflow. These were discarded because the similarity scores produced for pairs of images were not discriminative enough.

5 Demo

With the aim of testing our chosen method on a set of fingerprints, we created a Python notebook that exposes a simple demo of matching two fingerprints selected by the user from our test set, which was divided into probe and gallery. The subdivision between probe and gallery is contained in a JSON file.

Figure 31: Probe and gallery test set json file.

When the program is running, the user can choose the probe finger image, the gallery finger image and the method to use for the extraction of the features/minutiae through a multiple selection list. As regards SIFT and ORB, the keypoints of the fingerprints will be calculated by the program and subsequently the matching between the two will be shown through the drawMatches() OpenCV function. When the user selects MINDTCT the program will use the MINDTCT tool to extract the features and generate the values for the minutiae which will be saved in a text file called <NameFingerprint>.xyt. This file contains 4 values for each minutia: x,y coordinates, theta orientation, quality of the minutia; and one line for each minutia. <NameFingerprint>.xyt is the feature file format which will be later used by the BOZORTH3 matching algorithm to generate the similarity score between the two fingerprint images.

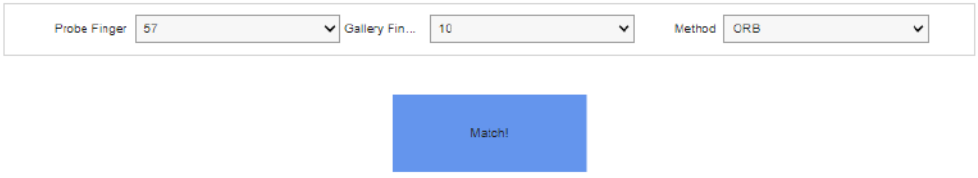

Figure 32: Demo example.

The user, after selecting probe, gallery and feature extraction method will click on the “Match!” button that will highlight the matches related to the chosen method. In the following images we can see how the demo works on SIFT, ORB and MINDTCT respectively.

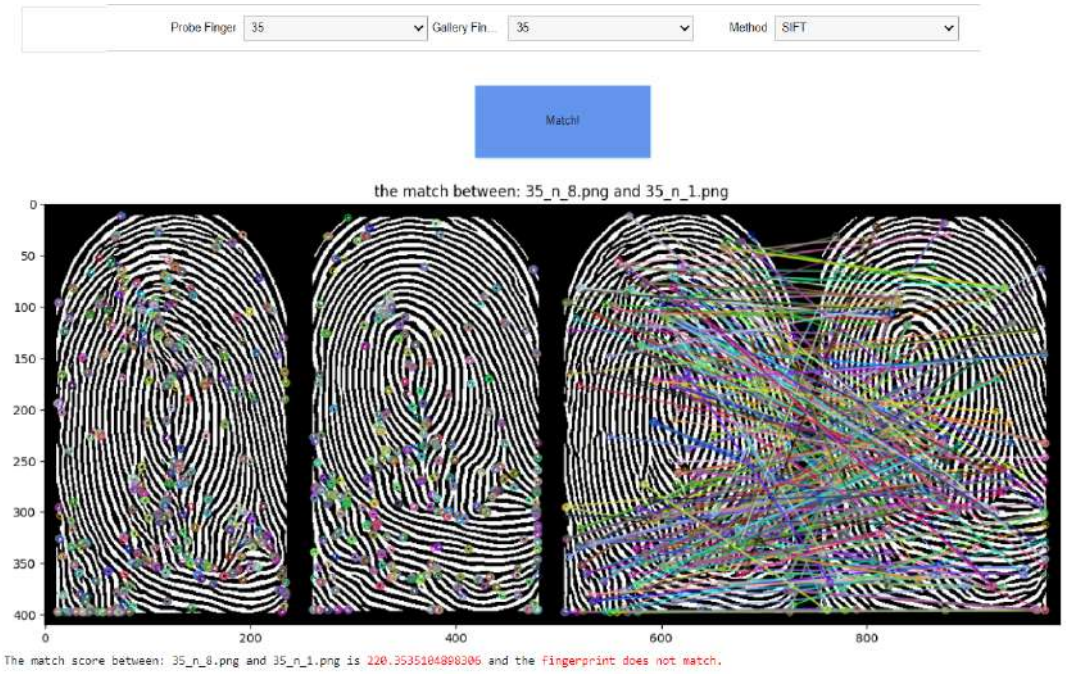

Figure 33: Demo example: SIFT matching.

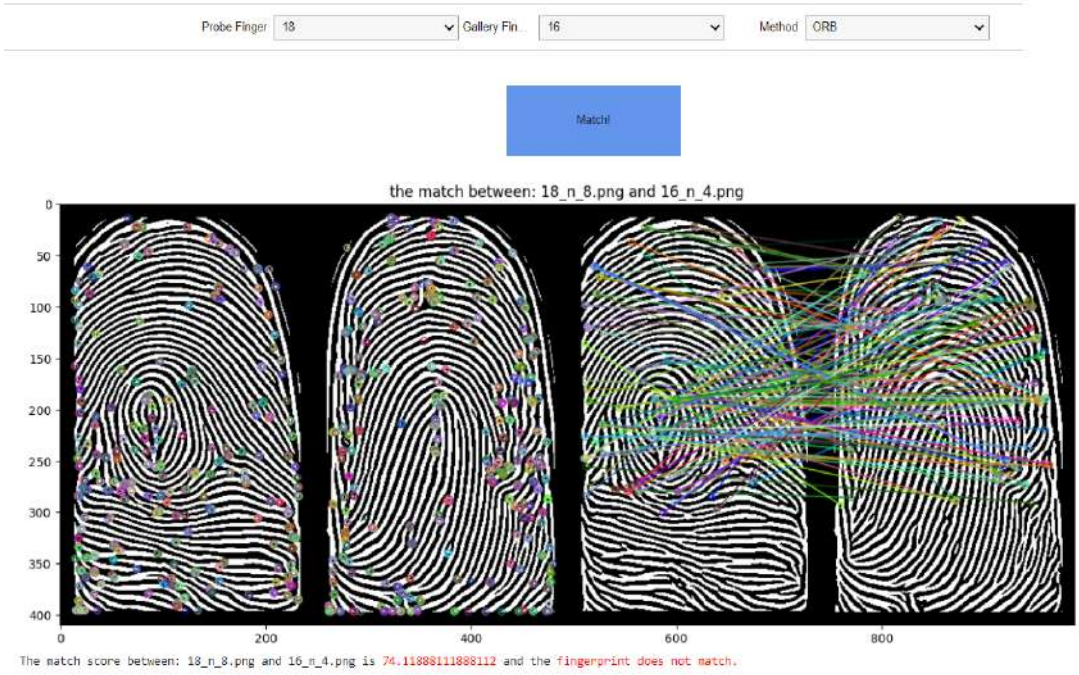

Figure 34: Demo example: ORB matching.

Figure 35: Demo example: MINDTCT features.

6 Conclusion 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, from classical computer vision methods to deep neural networks. In a nutshell, this project consisted in implementing a contactless fingerprint recognition task. One of the first problems we encountered was to extract a fingerprint from a finger photo that had a quality as close as possible to a livescan. In fact, this procedure required a long pipeline of preprocessing steps, such as the finger segmentation, ROI cropping and image enhancement, which in turn included several computer vision procedures. Then we used the NBIS software, which includes the NFIQ2 tool that allowed us to assess the quality of the images. Specifically, for each fingerprint image we obtained a quality score by which we could discard subsets with lowest quality. It was also necessary to split the dataset into a training set and a test set to ensure that we could find the best threshold value putting ourselves in a real case scenario. We did a further division in probe and gallery sets: for the training set we have chosen to make a single division between the two sets and for the testing we did multiple divisions in order to get reliable results.
At this point, we faced the feature extraction step, so we compared different techniques that are based on:

- blobs and local keypoints: since the extraction of minutiae in low quality fingerprint images is problematic, these methods use other features that are easier to extract, but also less distinctive. In our case we used SIFT and ORB;
- minutiae: methods that extract the minutiae positions from the fingerprint and the orientation of their angle. An example of this approach is MINDTCT.

Through an all probe vs. all gallery approach, we simulated the open set identification task and we compared the results obtained with these techniques. We found that matching based on minutiae gives us better performance than methods based on extracting local keypoints. Finally, we can say that although it was a very complicated task, we managed to achieve very good performance on a subset of the ISPFDv1 dataset. As a possible future work the work could be scaled on the entire dataset, or the entire process of fingerprint recognition that we have implemented could be integrated in a real system capable of verifying a person's identity simply by taking a picture of its finger. This would facilitate the process of fingerprint acquisition as no specific tool would be needed. Actually, it would be sufficient to use the camera of a smartphone, a medium that nowadays everyone can afford and that allows to take high quality photos.

References

[1] M. Abdullah-Al-Wadud, Md. Hasanul Kabir, M. Ali Akber Dewan, and Oksam Chae. A dynamic histogram equalization for image contrast enhancement. *IEEE Transactions on Consumer Electronics*, 53(2):593–600, 2007.

[2] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *SIGGRAPH*, 26, 07 2007.

[3] A.M. Bazen and S.H. Gerez. Systematic methods for the computation of the directional fields and singular points of fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):905–919, 2002.

[4] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. volume 6314, pages 778–792, 09 2010.

[5] Raffaele Cappelli, Matteo Ferrara, and Davide Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2128–2141, 2010.

[6] Lin Hong, Yifei Wan, and A. Jain. Fingerprint image enhancement: algorithm and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):777–789, 1998.

[7] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[8] Kenneth Ko. User's guide to nist biometric image software (nbis), 2007-01-21 2007.

[9] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2023.

[10] David G Lowe. Distinctive image features from scale-invariant keypoints. volume 60, pages 91–110. Springer, 2004.

[11] Jannis Priesnitz, Christian Rathgeb, Nicolas Buchmann, Christoph Busch, and Marian Margraf. An overview of touchless 2d fingerprint recognition. *J. Image Video Process.*, 2021(1), feb 2021.

[12] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. volume 106, page 107404, 2020.

[13] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[14] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[15] Anush Sankaran, Aakarsh Malhotra, Apoorva Mittal, Mayank Vatsa, and Richa Singh. On smartphone camera based fingerphoto authentication. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–7, 2015.

[16] Elham Tabassi, Martin Olsen, Oliver Bausinger, Christoph Busch, Andrew Figlarz, Gregory Fiumara, Olaf Henniger, Johannes Merkle, Timo Ruhland, Christopher Schiel, and Michael Schwaiger. Nist fingerprint image quality 2, 2021-07-13 04:07:00 2021.

[17] Zhenqiang Ying, Ge Li, Yurui Ren, Ronggang Wang, and Wenmin Wang. A new image contrast enhancement algorithm using exposure fusion framework. pages 36–46, 07 2017.