Assignment 2: Fingerprint and Iris based Identification

Daniele Giannuzzi r0876755

1 Setting

A woman was found dead inside a hotel room. Some stab wounds show that there is a killer inside the hotel. However the murderer who entered the room 15 minutes before her death left his fingerprint on the doorknob. In Figure 1 his fingerprint is shown. We also know that 100 guests were in the hotel during the murder.

As a forensic team, we have a database where all their fingerprints are collected. In Figure 2 some of those is shown.

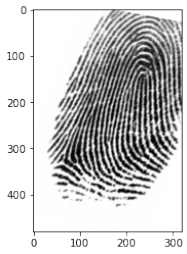

Figure 1: Murderer fingerprint

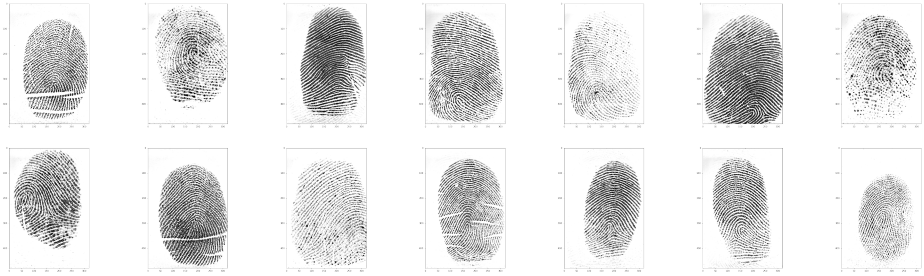

Figure 2: Guests fingerprint

Thanks to video surveillance records, we can also rely on a database of iris images to find the murderer. Perpetrator and some guests irises are represented in Figure 3 and Figure 4 respectively.

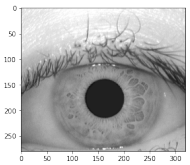

Figure 3: Murderer iris

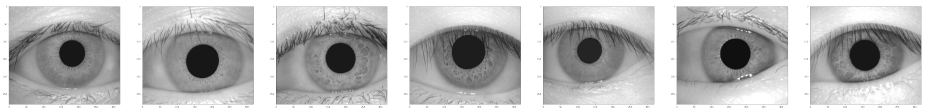

Figure 4: Guests iris

Since we want to associate a particular individual with an identity, we are in an identification scenario.

2 Fingerprint recognition

First of all, we can start our analysis evaluating information coming from the fingerprints data we have.

2.1 A baseline model

All we should do to evaluate the probability that a person of the group of 100 could be the murderer is to calculate the similarity score between the murderer fingerprint and each fingerprint in the database, using a similarity metric of choice. In this way we can construct a similarity table with which we can take a decision about who the killer can be.

As a baseline model, we can simply take the murderer fingerprint image and the other 100 and compute the pixels euclidean distance, or as done in the starting code, we can take the mean of the squared difference of the two images as a distance metric, and compute then the similarity as $\frac{1}{1+distance}$.

However how we can see from the Figure 5, this naive way of proceeding, comparing images on a pixel level, is too simple and not reliable. The similarity function we used is not a good metric to quantify the distance between two fingerprints since we are not taking into account the affine transformations. This results in a curve which is not so informative since the scores are not so different. For example, the highest match score 0.0170 belongs to subject 088. However, subject 081 is not far behind with a score of 0.0168.

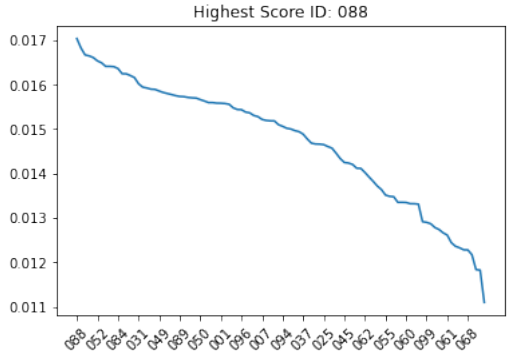

Figure 5: Baseline model: scores curve

2.2 Image enhancements

As one can observe from Figure 2 the fingerprint images from a guest to one another differ a lot, for instance: for the fingerprint position and orientation in the image; the intensity of the grey and thickness of the ridges; for some noise outside the image region of interest. This depends on the fingerprint acquisition, since for example a finger can be dry or wet, resulting in a different fingerprint image. However to extract the same information from equal fingerprints for which the acquisition was done at different times, we need images with similar conditions. Therefore we need first of all to improve the fingerprint images.

a. To enhance fingerprint images we can use several techniques. In this case we used the code developed by Utkarsh Deshmukh. It uses a Gabor filterbank which takes as input, to set the parameters of θ and f, the gradient-based orientation and the local frequency estimation. The second row of Figure 6 represents the result of this process.

b. Then to focus only on the region of interest, disregarding all the noise around, segmentation is performed. The resulting mask is shown in the third row of the Figure 6. Since the region-based segmentation methods are preferred over edge-based segmentation methods in case of a noisy image, instead of the initial segmentor implemented in the code, a region-based image segmentation algorithm has been used. It consists of morphological operations such as dilation, opening and closing, then given a threshold markers are computed and passed to the watershed segmentation algorithm. Finally the remaining holes are filled.The comparison of this approach with the other segmentation methods will be discussed later.

The final result of the entire enhancement process is represented in the final row of the Figure 6.

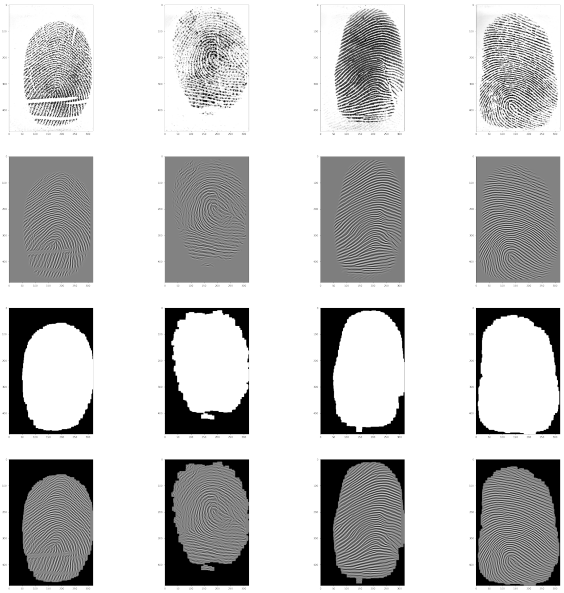

Figure 6: Fingerprint images enhancement. 1^{st} row: original fingerprint images; 2^{nd} row: gabor filtered fingerprints; 3^{rd} row: fingerprints segmentation; 4^{th} row: final fingerprint image enhanced and segmented

2.3 Feature extraction

In literature, before the advent of Deep Learning, the main strategy used for fingerprint recognition was based on matching keypoints, called minutiae. Many types of minutiae exist, however ridge endings and bifurcations seem to be the most meaningful. Nevertheless, in the approach used in this case, established methods developed in Computer Vision to extract keypoints and the relative descriptors are used. They do not necessarely extract the previously described minutiae from the image, but points of interest, sometimes not recognisable by humans, which can be discriminative from one fingerprint to one another.

Here we use the SIFT (Scale-invariant feature transform) to detect keypoints, since in the optional point discussed later, it seems to perform better than the ORB (Oriented FAST and Rotated BRIEF) originally implemented. Their differences will be argued afterwards. Moreover since many false keypoints are generated at the edge of the foreground mask, since ridges seem to terminate due to the clipping, we remove these by a morpholigical erosion of the foreground mask and deleting the keypoints outside. The result of this process is shown in Figure 7 where the keypoints founded by the detection algorithm are highlighted by the green circles.

A Brute-Force matcher is then used to compare the keypoints of image 48 and 17 of the fingerprints database to figure out if the keypoints founded are meaningful. This matcher takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. Then the closest one is returned. The effects of its application can be observed in Figure 8.

Figure 7: Fingerprint keypoints

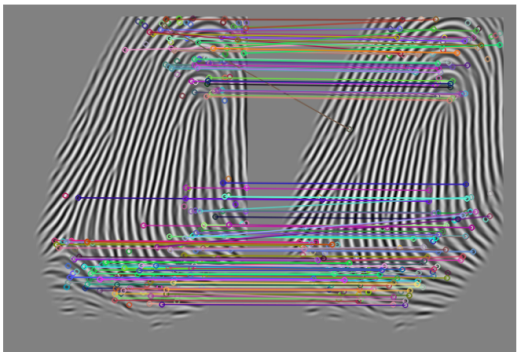

Figure 8: Fingerprints matches

As we can see, not all matches are correct since we are using a brute force matcher with euclidean distance (since we use SIFT), and we can not be sure that the local keypoints correspond to the same landmark. However, using the ORB with hamming distance produce even much more false matches.

2.4 Local similarity

Now that we have matching keypoints, we can sort these based on the distance. Then we can use the mean of the first N distances, where $N = 22$. Now we can construct the similarity table and evaluate the best scores.

Rank	ID	Scores
1	049	0.11483
2	056	0.05039
3	022	0.04419
4	048	0.03987
5	018	0.03662
6	076	0.03573
7	081	0.02492
8	095	0.02401
9	065	0.02401
10	015	0.02401

Table 1: First 10 fingerprint local similarity scores

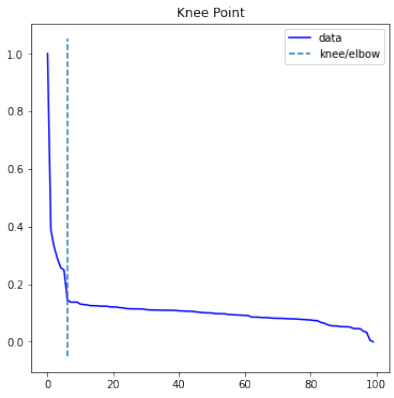

Figure 9: Knee point detection for fingerprint local similarity scores

Since the first subjects scores are very close as one can notice from the Table 1, a good idea is to to take a look at the knee point for a score threshold, normalizing the scores. The Figure 9 shows the knee point detection, which finds $threshold = 0.14611$ and identifies that the best 6 matches are all pretty close.

2.5 Global similarity

As the local similarity scores are not so discriminative to clearly find the culprit, we can imagine that using a global similarity metric could help to discriminate better.

First of all, to compute global features we need to align the fingerprint images.

The method used to find the trasformation matrix useful to align the images is the $estimateAffinePartial2D$ present in the $OpenCV$ library. It makes use in the implementation of the RANSAC (RANdom SAmple Consensus) method. It is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers. The similarity transformation implemented here, iteratively determine the minimal set of matching points (inliers) to optimally align all other points as well, taking care of outliers at the same time. In this way only the keypoints considered good are used to find the transformation.

An example can be seen in Figure 10 where one can observe the inliers (with their matches) used for the calculation of the transformation matrix necessary for the alignment of the fingerprint on the left. As one can notice, all the wrong matches (i.e. diagonal lines) present in Figure 8 are no more present. This means that the outliers detection works and the alignment is reliable.

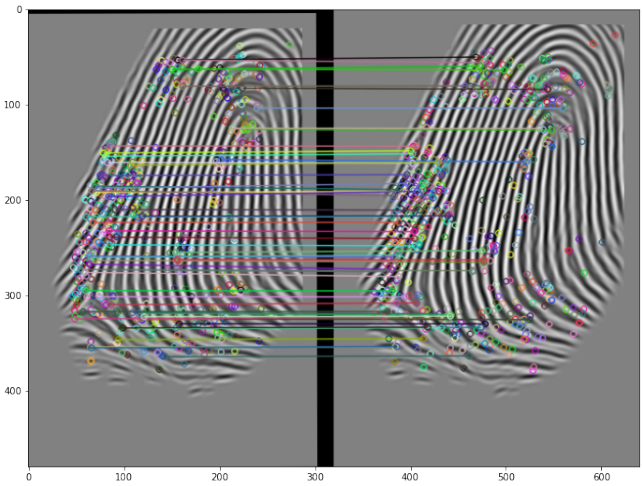

Figure 10: Inliers matches and alignment for global similarity

Once the images are aligned, a global feature similarity function can be chosen. We now consider the reduced set of key-points (inliers) and we calculate the euclidean distance between the coordinates of these. Then, only the first K smaller distances are taken into account to compute the mean distance, useful as a distance metric for two fingerprint images. The similarity metric can be easily retrieved by taking the reciprocal.

The experimental results are reported in Table 2 and Figure 11. Despite a global similarity method was used, the same problem persists: the knee is founded for a $threshold = 0.147$ after the scores normalization. This means that again we can not clearly decide who is the identity of the impostor, since the knee highlights that the first 13 best fingerprint matches are too close in score.

Rank	ID	Scores
1	050	10.007
2	049	9.062
3	022	8.781
4	056	6.832
5	076	6.701
6	018	6.202
7	048	5.653
8	003	3.549
9	045	3.261
10	087	3.186

Table 2: First 10 fp global similarity scores

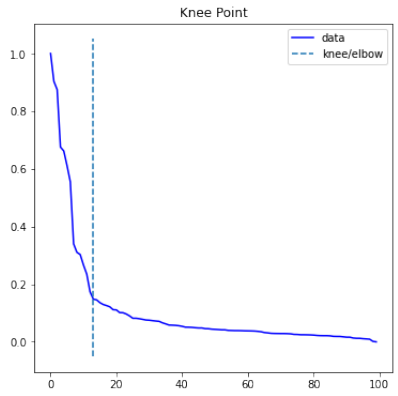

Figure 11: Knee point detection for fingerprint global similarity scores

2.6 Hybrid features

Given the problems of a lack of clear distinction, we can hope that an hybrid setup could be able to give us better results.

It is straightforward, since we simply need to fuse the scores. First of all, being similarity scores, they are already normalized between 0 and 1. Therefore we just need a way to take both into account. We have several kinds of mean at our disposal: mean, weighted mean, geometric mean, harmonic mean.

The harmonic mean is chosen. Since the reciprocals of the values of the variable are involved, it gives greater weightage to smaller observations and as such is not very much affected by one big observation.

The results shown in Table 3 and Figure 12 also with this approach are not satisfying. We have a knee at 8 with a $threshold = 0.260$ so the same problem persists.

Rank	ID	Scores
1	049	0.16274
2	056	0.08200
3	048	0.07162
4	022	0.07003
5	018	0.06536
6	076	0.06401
7	020	0.05687
8	053	0.05615
9	095	0.05393
10	013	0.05334

Table 3: First 10 fp hybrid similarity scores

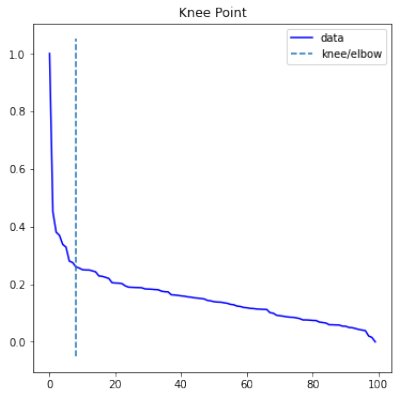

Figure 12: Knee point detection for fingerprint hybrid similarity scores

However, if we print out the first 7 fingerprint images in the ranking as done in Figure 13 we clearly notice why we always encountered the same problem. The first 6 fingerprints are simply the same! The murderer must have hacked the database and replicated some of the images.

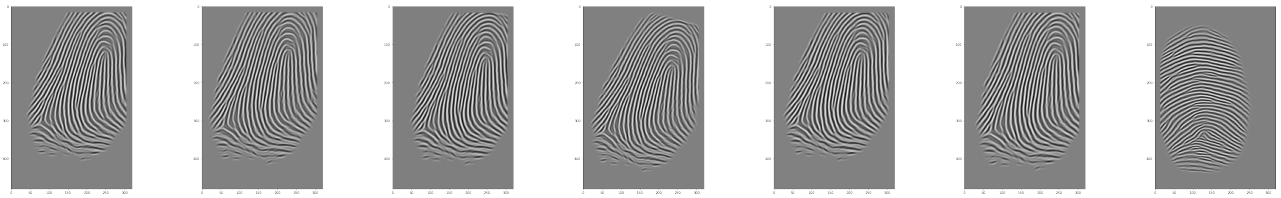

Figure 13: First 7 fingerprint images in the hybrid ranking

3 Iris recognition

Thankfully even if the fingerprint database has been hacked, we also have the iris database to exploit to find the culprit.

With respect to the steps we had for the fingerprint recognition, what is really different in the approach we are going to follow is the image preprocessing. Indeed the iris segmentation is crucial in iris biometric systems: inaccuracies in localising the iris can negatively impact the performances, also in this case in which we are going to use the keypoints and descriptors instead of the well established technique based on IrisCodes. Moreover, unlike the fingeprints, here we can have eyelashes and eyelids which could occlude the iris region. Also the circular shape of the iris is a further difficulty that needs to be resolved.

3.1 Image enhancement

Image preprocessing for the iris follows several steps:

a. iris segmentation: to segment the iris region from the eye image we use open source code based on Daugman's intefro-differential, we also normalize iris region by unwraping the circular region into a rectangular block of constant dimensions;

b. iris image enhancement: similar to the fingerprint section we enhance the images using a gabor filter bank from $OpenCV$. How the parameters of the gabor filter influence the enhanced image will be discussed later for

the optional task. The results of these first steps can be observed on the first row of Figure 14

c. eyelids and eyelashes segmentation: we remove with segmentation the parts of eyelids and eyelashes that occlude the iris, as can be seen on the second row of Figure 14

The final result of iris images preprocessing is represented in the last row of the 14.

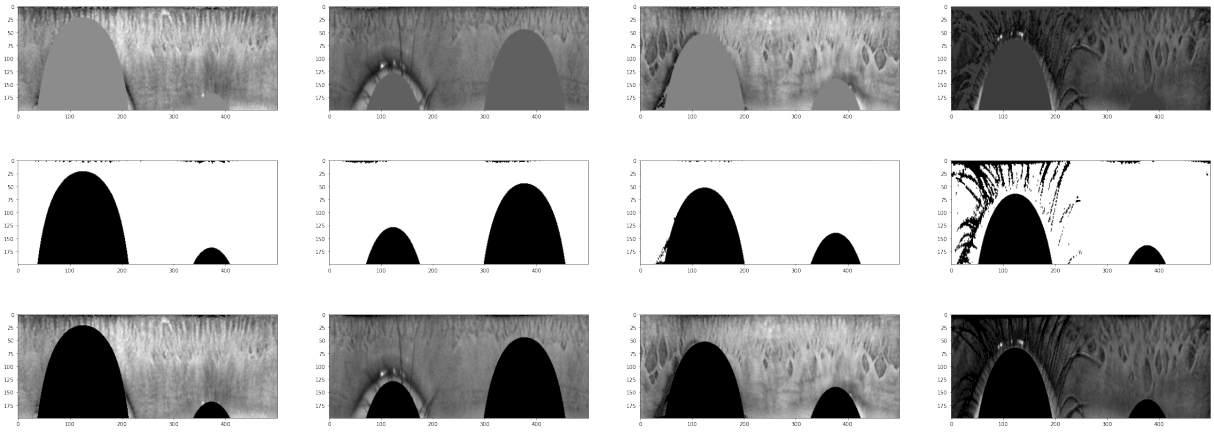

Figure 14: Iris images enhancement: 1st row: iris images after segmentation and enhancement; 2nd row: eyelids segmentation; 3rd row: final result of iris images preprocessing

3.2 Local similarity

Given the enhanced iris images, we can use the same strategy adopted for the fingerprint. Also in this case we use $N = 22$ as number of reliable matches from which calculate the similarity score. The experimental results are reported in Table 4 and Figure 15. The knee is at 8, with the $threshold = 0.4081$. The same problem had with the fingerprints seems to arise.

Rank	ID	Score
1	049	0.11483
2	056	0.05039
3	022	0.04419
4	048	0.03987
5	018	0.03662
6	076	0.03573
7	081	0.02492
8	095	0.02401
9	065	0.02401
10	015	0.02401

Table 4: First 10 iris local similarity scores

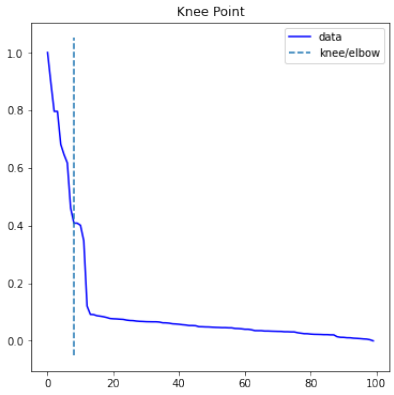

Figure 15: Knee point detection for iris local similarity scores

3.3 Global similarity (optional)

Trying to see if something changes, we construct the global similarity table as it was done for the fingerprint images, aligning the images and using the euclidean distance to calculate the coordinate distance of the inlier keypoints. The similarity table is calculated by the reciprocal of the distances.

As one can observe from Table 5 and Figure 16, there is not clear information to distinguish the murderer.

3.4 Hybrid similarity (optional)

We try to fuse again the local and global method at the score level, using the harmonic mean. The results are shown in Table 6 and Figure 17.

Not a huge difference with the previous strategies can be noticed. This means that probably the perpetrator has hacked also the iris database. Indeed the first 7 iris images for hybrid scores are represented in Figure 18, and as happened for the fingerprints they are all referring to the same iris.

Rank	ID	Score
1	056	0.12899
2	083	0.12725
3	036	0.12453
4	063	0.12193
5	093	0.10199
6	091	0.10199
7	025	0.08975
8	062	0.08239
9	010	0.07615
10	003	0.07462

Table 5: First 10 iris global similarity scores

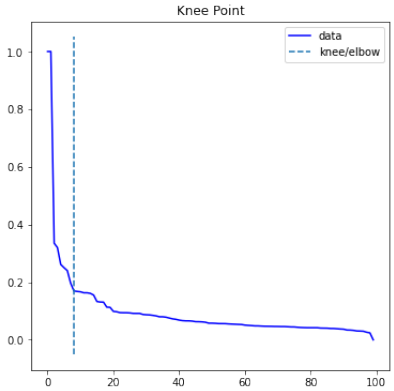

Figure 16: Knee point detection for iris global similarity scores

Rank	ID	Score
1	049	0.16274
2	056	0.08200
3	048	0.07162
4	022	0.07003
5	018	0.06536
6	076	0.06401
7	020	0.05687
8	053	0.05615
9	095	0.05393
10	013	0.05334

Table 6: First 10 iris hybrid similarity scores

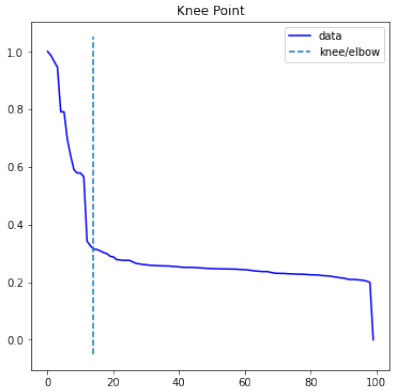

Figure 17: Knee point detection for iris hybrid similarity scores

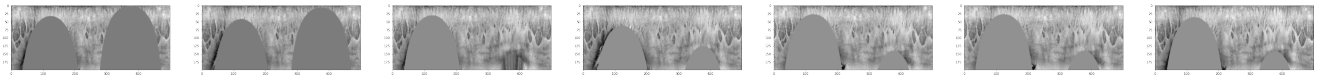

Figure 18: First 7 iris images in the hybrid ranking

4 Multimodal System

To solve the murder case we can fuse the iris and fingerprint biometric systems on the score level, hoping that the hacked similar fingerprints and irises correspond to different identities.

To do that we can simply normalize the fingerprint and iris scores from 0 and 1, and then fuse them with a simple mean. The final scores and rankings are reported in the Table 7. The first identity in the ranking is quite far from the others which have scores very close to each other. Therefore, we can be quite sure that $ID = 056$ with the $score = 0.72544$ is the murderer.

Rank	ID	Score
1	056	0.72544
2	049	0.60465
3	036	0.59633
4	083	0.57491
5	091	0.52041
6	063	0.51874
7	093	0.46413
8	025	0.38487
9	003	0.36507
10	010	0.36417

Table 7: First 10 multimodal similarity scores

5 Optional tasks

All the implementations are at the end of the notebook, ordered task by task. The only exception is the point E, for which only the parameters of the affine transformation are changed in the code, and the point G for which the segmentors are implemented in $Assignment2/src/fprmodules/enhancement/image\_enhance$ py file and just imported in the jupyter notebook.

5.1 A. Different KeyPoint detectors for fingerprint

Several KeyPoint detectors were tried to figure out which one can perform better for the fingerprint recognition task:

- ORB: Oriented FAST and Rotated BRIEF is an algorithm developed because SIFT and SURF are patented algorithms. It generally performs as good as SIFT but it is faster. It was used with default parameters.
- SIFT: Scale-Invariant Feature Transform, it finds features which are invariant to image scale and rotation and robust against affine distortion and noise. It is one of the best keypoint detectors, while its main drawback is the computational slowness. Since with the default parameters it detects very a low number of keypoints, we have set $edgeThreshold = 30$, $contrastThreshold = 0.05$, in order to get approximately the same number of keypoints of the two other methods.
- BRISK: Binary Robust Invariant Scalable Keypoints, as suggested by the name, the method is rotation as well as scale invariant, achieving performance comparable to SIFT and SURF while dramatically reducing computational cost. Default parameters were used.

The Figure 19 clearly shows that SIFT performs better than the two other methods, for this particular way of enhancing the fingerprint images. Indeed, even if it is not capable in detecting minutiae every time, as well as ORB and BRISK, its keypoints and descriptors seem to generate better matches at a local level. On the other hand, BRISK seems to be the worst, with a lot of false matches, highlighted by the high number of diagonal matching lines.

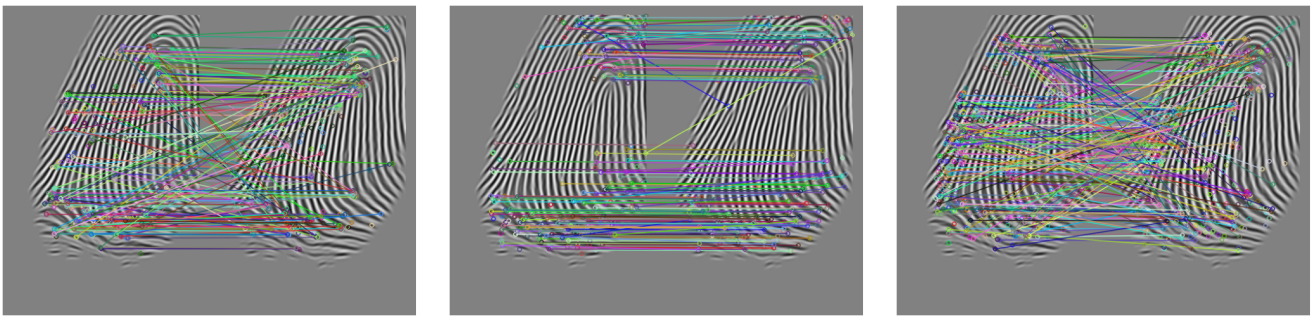

Figure 19: Matching comparison of different KeyPoint detectors, in order: 1. ORB; 2. SIFT; 3. BRISK

5.2 B. Playing with gabor filterbank parameters for iris enhancement

The parameters used by default in the code for gabor filter method are: $ksize = 5$, $sigma = 4$, $theta\_range = np.arange(0, np.pi, np.pi/16)$, $lambda = 10$, $gamma = 0.5$, $psi = 0$. Since we want to analyze the behaviour of the enhancement with respect to the different parameters, we vary the parameters one by one. The theta division in the caption of the Figure 22 refers to the variation to the value of 16 used in the original code for the parameter $theta\_range$.

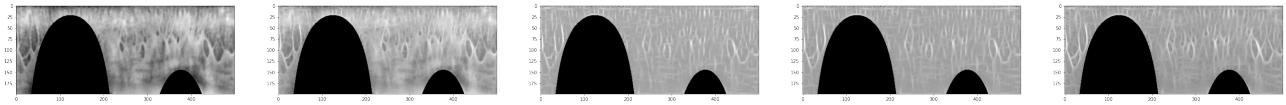

Figure 20: Gabor filter enhancement with ksize = [5, 9, 13, 17, 21]

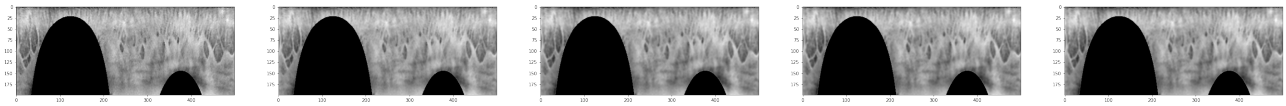

Figure 21: Gabor filter enhancement with sigma = [0.5, 1, 4, 16, 32]

Given these comparisons, the most important parameters to set to have a better enhancement seem to be: $ksize$: for high values the image enhancement worsens; $sigma$: we can control the sharpness of the image;

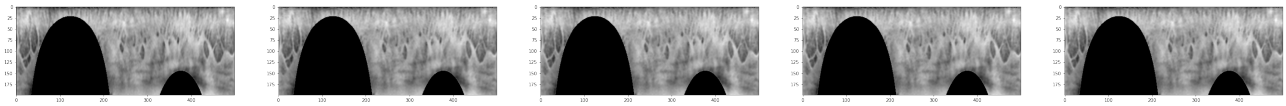

Figure 22: Gabor filter enhancement with theta division = [0.7, 2, 32, 64, 128]

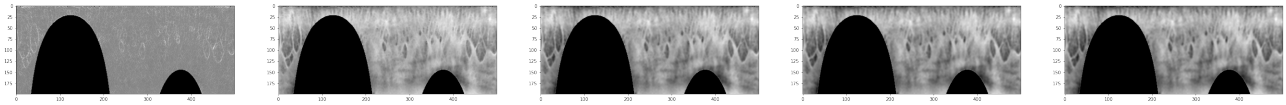

Figure 23: Gabor filter enhancement with lambda = [4, 7, 10, 15, 25]

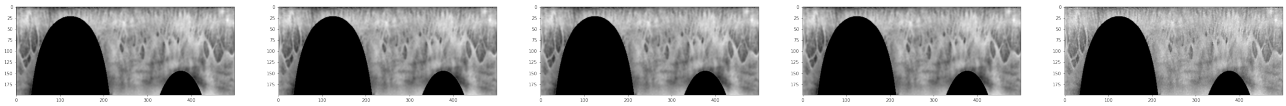

Figure 24: Gabor filter enhancement with gamma = [0.1, 0.3, 0.5, 0.7, 100]

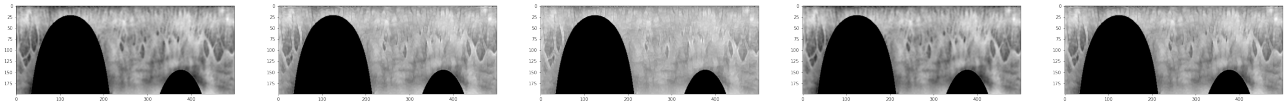

Figure 25: Gabor filter enhancement with psi = [0, 1, 2, 3, 4]

$theta\_division$: for higher value an greater range of orientations are tested; $lambda$: low values worsen the image. The founded values of all the parameters which seem to make the scores of the tables more discriminative are: $ksize = 3$, $sigma = 1$, $theta\_range = np.arange(0, np.pi, np.pi/128)$, $lambda = 10$, $gamma = 0.5$, $psi = 1$.

5.3 D. Comparison of local, global and hybrid features for iris

This optional point has been already discussed in the Iris Recognition paragraph.

5.4 E. Attempt to improve the registration procedure

To improve the registration (i.e. alignment) procedure it was enough to simply try other values for the input parameters of the $estimateAffinePartial2D$.

The best parameters values found along with their description are:

- $confidence = 0.99$: it is the confidence level, between 0 and 1, for the estimated transformation. Low values result in an incorrectly estimated transformation
- $ransacReprojThreshold = 9.0$: maximum reprojection error in the RANSAC algorithm to consider a point as an inlier
- $maxIters = 9000$: the maximum number of robust method iterations
- $refineIters = 10$: maximum number of iterations of refining algorithm (Levenberg-Marquardt)

5.5 G. Attempt to improve the fingerprint segmentation

Several attempts were made to improve the fingerprint segmentation. Here a list of the strategies used with their description.

- Default improved: to improve the segmentation of the algorithm already implemented in the code, in the $image\_enhance$ method the $thresh$ parameter was set from 0.1 to 0.2.
- Canny: to increase the accuracy of the segmentation process we use thresholding for enhancing the image by increasing the contrast for better edge detection, and morphological operations such as dilation and opening. After finding the edges of the object with the Canny algorithm, the segmentation mask is found using the $drawContours$ method of $OpenCV$.
- Kmeans: with this algorithm we segment the image selecting K points and assigning them a cluster. Clearly in this case $K = 2$, since we just want to have a mask. Morphological operations are used as well.
- Region-based: already described before, it makes use of morphological operations, markers and watershed strategy of segmentation.

As we can observe from the Figure 26 the Default and the Region-based algorithms seem to work better then the others for the first fingerprint, while the Kmeans and Region-based do a better job on the third fingerprint.

The last fingerprint is quite easy to segment for all of them, while the second one is very hard to precisely segment, so we can state that they all do a good job for this finger. Given these considerations, Region-based seems to be the best performing. Numerically, it showed to perform in the generation of the scores and knees, practically the same way as the improved Default algorithm.

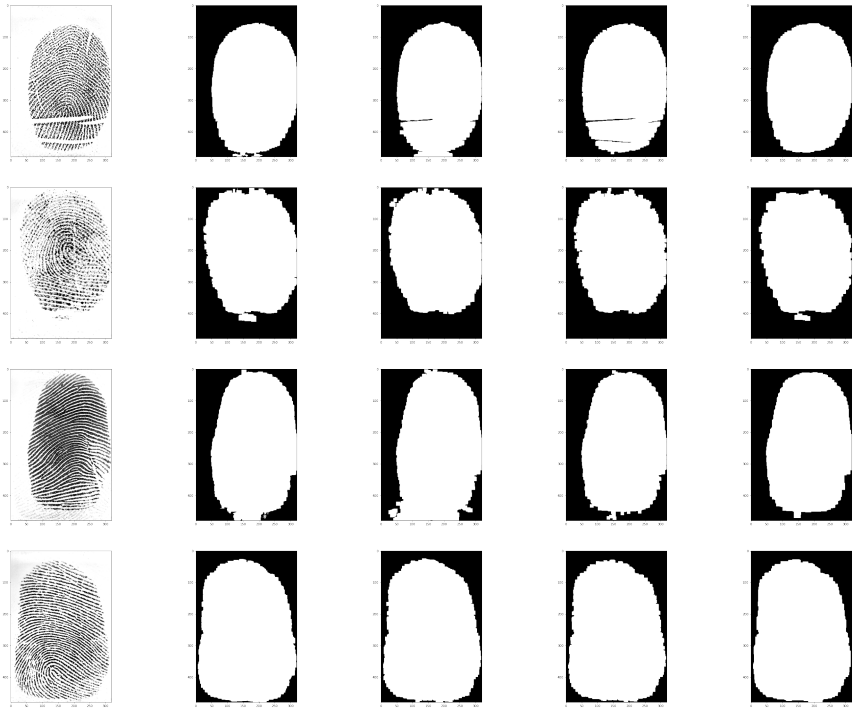

Figure 26: Different algorithm of segmentation in order: 1. Default improved; 2. Canny; 3. Kmeans; 4. Region-based. The fingerprint considered are the first 4 in the guests database